WRC RESEARCH REPORT NO. 73

ILLINOIS STORM SEWER SYSTEM SIMULATION MODEL: USER'S MANUAL

A. Suha Sevuk
Ben Chie Yen
Department of Civil Engineering

and

Gordon E. Peterson
Department of Computer Sciences
University of Illinois at Urbana-Champaign

T E C H N I C A L R E P O R T
Project No. B-043-ILL

The work upon which this publication is based is supported by funds provided by the U.S. Department of the Interior as authorized under the Water Resources Research Act of 1964, P.L. 88-379
Agreement No. 14-31-0001-3078

UNIVERSITY OF ILLINOIS
WATER RESOURCES CENTER
2535 Hydrosystems Laboratory
Urbana, Illinois 61801

October 1973

ABSTRACT

ILLINOIS STORM SEWER SYSTEM SIMULATION MODEL: USER'S MANUAL

The Illinois Storm Sewer System Simulation Model is a mathematical model for sewer design and flow prediction utilizing the Saint Venant equations to route unsteady flows through tree-type sewer networks. An overlapping segment scheme is used in the numerical solutions to account for the backwater effects and mutual influences of the sewers and junctions. The program is written in PL/1 and assembler Language and can be executed on most large IBM 360 and 370 systems. User oriented information is provided in this report. An example on sewer design is also given.

Sevuk, A. Suha, Yen, Ben Chie, and Peterson, Gordon E.
ILLINOIS STORM SEWER SYSTEM SIMULATION MODEL: USER'S MANUAL
Technical Report to Office of Water Resources Research, Department of the Interior, Washington, D.C., Research Report No. 73, Water Resources Center, University of Illinois, Urbana, Illinois, October 1973, vii + 168 p.
KEYWORDS--*computer models, drainage systems, flood routing, hydraulics, *hydraulic design, hydrograph, mathematical model, open-channel flow, *simulation, *storm sewers, unsteady flow, *urban runoff

FORWARD

The Illinois Storm Sewer System Simulation Model (ISS Model) has been developed as a part of the results of the research project "Methodologies for Flow Prediction in Urban Storm Drainage Systems" which was directed by Professor B. C. Yen and sponsored by the University of Illinois and the Office of Water Resources Research under the Agreement No. 14-31-0001-3078, Project No. B-043-ILL. The results of this project have been summarized in Research Report No. 72 of the Water Resources Center of the University of Illinois.

Since the development of the ISS Model and subsequent report of the Model in Dr. A. S. Sevuk's dissertation, requests have been received concerning the availability of the computer program of the Model. In view of the complex nature of the program it was decided that a manual should be written and published for the convenience of the users. Furthermore, arrangement has been made with the Digital Computer Laboratory that tapes can be provided at cost and requests should be sent through the second author.

The ISS Model reported herewith is a slightly improved model from that reported in the first author's dissertation. The third author acting as a student helper programmer contributed greatly in the programming details.

CONTENTS

LIST OF FIGURES

NOTATION

A = cross-sectional area of flow in sewer;

A_j = cross-sectional area of junction;

B = surface width of flow in sewer;

C^+ = forward characteristic;

C^- = backward characteristic;

D = diameter of sewer;

$F = \sqrt{gB/A}$;

f = Darcy-Weisbach resistance coefficient;

$f_i(\ )$ = function;

g = gravitational acceleration;

h = water depth in junction;

k = effective roughness of sewer;

L = length of sewer;

Q = discharge;

$q(\ )$ = function representing time variation of inflow;

Q_b = base flow rate in sewer;

Q_j = direct junction inflow;

Q_p = peak discharge;

R = hydraulic radius;

$\mathbb{R} = vR/\nu$, Reynolds number;

$\mathbb{R}^*$ = threshold Reynolds number which separates hydraulically smooth and rough regions;

S_o = sewer slope;

S_f = friction slope;

s = storage;

T = duration of flood wave;

t = time;

V = velocity of flow;

V_o = steady uniform flow velocity in sewers at half-full;

W_1 = sum of known terms in forward characteristic equation (Eq. 31a);

W_2 = sum of known terms in backward characteristic equation (Eq. 31b);

x = distance along longitudinal direction of sewer;

y = depth of flow;

y_c = critical flow depth;

Z = height of drop at exit of sewer;

ν = kinematic viscosity;

θ = ratio of computational grid sizes ($\Delta t/\Delta x$); and

τ = time lag.

1. INTRODUCTION

Rapid urbanization of the land in the United States as well as elsewhere in the world together with increasing public concern of the urban living environment necessitates extensive construction, improvement, and proper operation and maintenance of urban drainage systems. Consequently, improved design and operation methodologies for urban storm sewer systems are most desirable. The Illinois Storm Sewer System Simulation Model (ISS Model) has been developed to serve a dual purpose; namely, (a) quantitative prediction of flow conditions at various locations within an existing sewer system; and (b) systematic hydraulic design of the size of the sewers of a new drainage system, or of extensions and modifications to an existing sewer system, with prescribed design inflows.

The hydraulic and mathematical development of the ISS Model has been reported elsewhere (2, 5)*. This report is a user's manual for the computer program implementing the Model. The purpose of this manual is to enable researchers and engineers to use the ISS Model in an effective manner with no more than a fundamental understanding of how digital computers accept input data.

The sewer networks simulated in the ISS Model are tree-type systems shown schematically in Fig. 1. For a junction with more than three sewers, only three of them are considered for direct backwater effects; others are regarded as a source of flow directly into the junction shown as Q_7 in Fig. 1. No pressurized flow is considered. The Model utilizes the St. Venant equations together with the compatibility conditions at the junctions to describe mathematically the flow.

In order to provide a proper background for the prospective user of the computer program, a brief review of the mathematical model, the governing

*Numerals in parentheses refer to corresponding entries in REFERENCES.

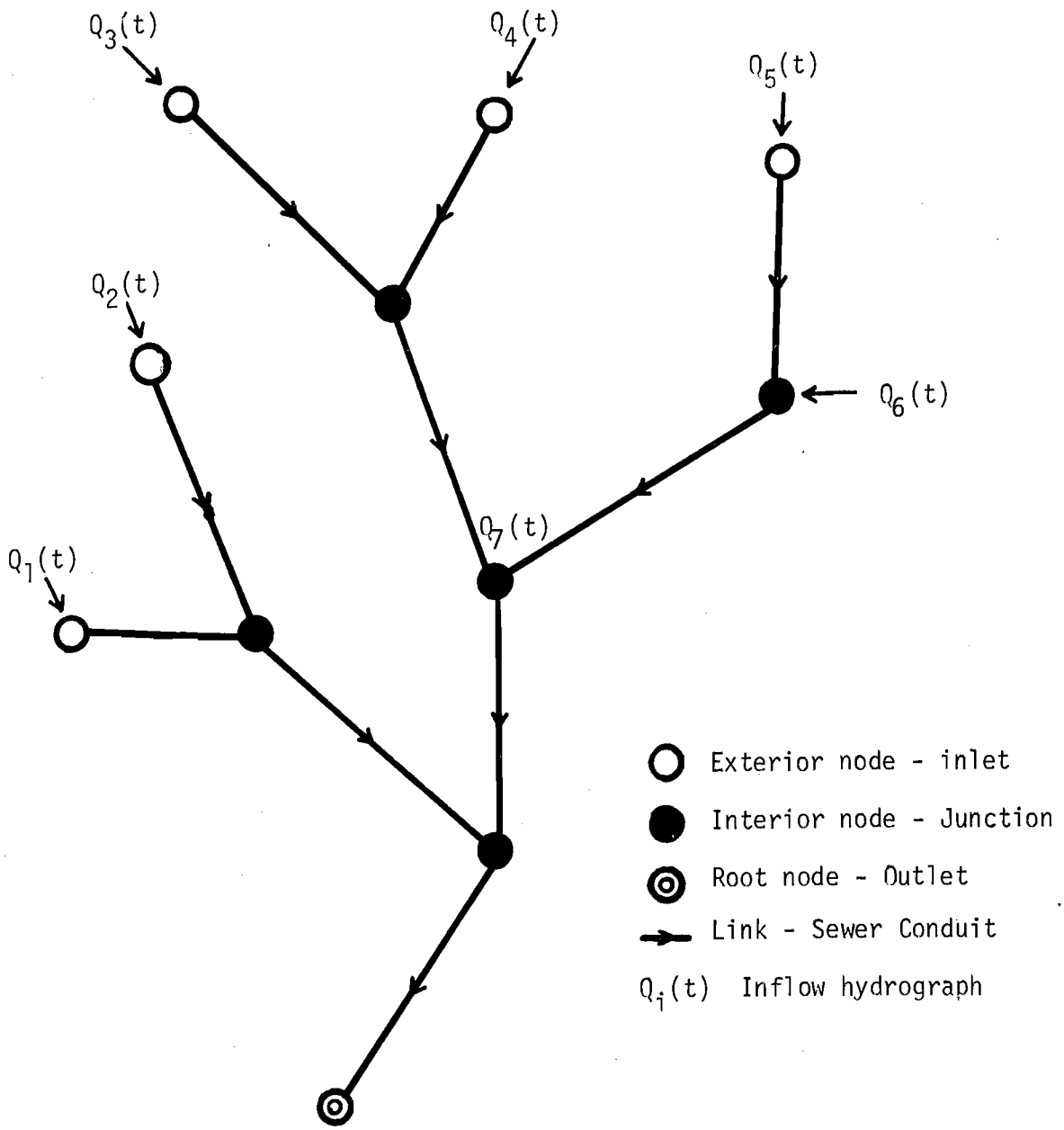


FIG. 1. TYPICAL TREE TYPE NETWORK

equations, and their numerical solution procedure is presented in Chapters 2 and 3. Those who are not interested in the hydraulics and solution method background can omit reading these two chapters. Conversely, those who are interested in more details on this aspect may refer to other reports (2, 5). In Chapter 4 general information concerning the computer program and requirements on the computer system are outlined. The input data preparation and the output from the ISS Model are described in Chapter 5.

The information contained in Chapters 4 and 5, together with the distribution tape of the computer program which is described in Chapter 7, is believed to be sufficient for the implementation of the ISS Model without the assistance of a programmer. The technical details of the program is further documented in Chapter 6 for those who are familiar with computer applications. The information given in Chapter 6 together with the program listing in Appendex A and B is believed sufficient for those who are interested in extension and modification of the ISS Model to suit his specific needs. In this latter case the assistance of an experienced programmer is strongly recommended to avoid unnecessary computing expenses.

2. MATHEMATICAL MODEL

2.1. Governing Equations

2.1.1. One-dimensional Unsteady Free-Surface Flow Equations

In the ISS Model the one-dimensional equations for gradually varied, unsteady, free-surface flow are used to describe the sewer flows. They are commonly known as the St. Venant equations (1, 4, 5)

$$B \frac{\partial y}{\partial t} + BV \frac{\partial y}{\partial x} + A \frac{\partial V}{\partial x} = 0 \tag{1}$$

$$\frac{\partial V}{\partial t} + V \frac{\partial V}{\partial x} = gS_o - g \frac{\partial y}{\partial x} - gS_f \tag{2}$$

in which A, B, and y are the cross-sectional area, water surface width, and depth of the flow in the sewer, respectively; V is the cross-sectional mean velocity; x is the distance along the longitudinal direction of the sewer; t is time; g is the gravitational acceleration; and S_o and S_f are the sewer bottom and friction slopes, respectively.

The friction slope, S_f, in Eq. 2 is evaluated by using the Darcy-Weisbach formula

$$S_f = f \frac{V^2}{8Rg} \tag{3}$$

in which f is the frictional resistance coefficient and R is the hydraulic radius. The value of f is approximated by that given in the Moody diagram for steady uniform flows (5). Since laminar flow rarely occurs in storm sewers, only turbulent flow is considered in the present mathematical model. Thus, for hydraulically smooth conduits the Blasius formula applies,

$$f = \frac{0.223}{\mathbb{R}^{0.25}} \tag{4}$$

for $\mathbb{R} < 4 \times 10^5$ where

$$\mathbb{R} = \frac{VR}{\nu} \tag{5}$$

is the Reynolds number in which ν is the kinematic viscosity of the fluid. For fully developed turbulent flow in hydraulically rough conduits

$$\frac{1}{\sqrt{f}} = 2 \log \frac{2R}{K} + 1.74 \tag{6}$$

in which K is a length measure of the surface roughness. In sewers hydraulically smooth-boundary flow rarely occurs with $\mathbb{R} > 10^6$. Hence if the Blasius formula (Eq. 4) is assumed to apply to higher Reynolds number and the transition between smooth and rough boundaries is neglected (5), the threshold Reynolds number, $\mathbb{R}^*$, which separates the hydraulically smooth and rough regions can be determined by eliminating f from Eqs. 4 and 6 to yield

$$\mathbb{R}^* = 0.633 \left(\log \frac{2R}{K} + 0.87\right)^8 \tag{7}$$

In the Model the value of f is computed from Eq. 4 or Eq. 6 depending on whether $\mathbb{R}$ is greater or less than $\mathbb{R}^*$.

2.1.2. Compatibility Equations at Junctions

In establishing a junction routing procedure, two options are provided depending on the relative size of the junction box.

(A) Point-Type Junction.

For a point-type junction the junction box is assumed to be represented by a single confluence point with no storage capacity. The net discharge into the junction is therefore zero at all times, i.e.

$$Q_1 + Q_2 + Q_j - Q_3 = 0 \tag{8}$$

in which the subscripts 1, 2 and 3 identify the joining sewers at the junction (Fig 2); and $Q_j = Q_j(t)$ represents the direct temporaly variable water inflow into, or the pumpage out from the junction box, if any.

Since the junction is considered as a point, common water surface exits for all the joining sewers. Thus, for the junction shown in Fig. 2a, and for subcritical inflows in the sewers i = 1, 2,

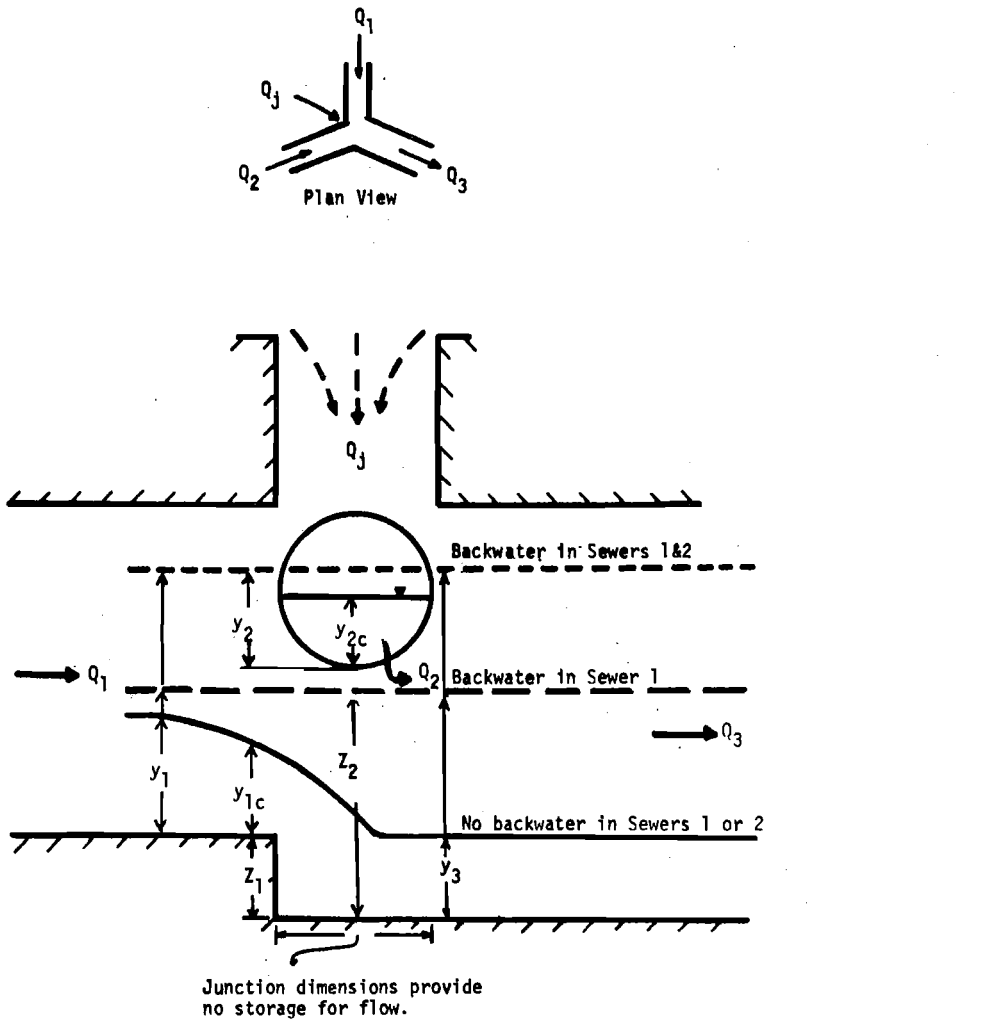
$$y_i = y_{ic} \qquad \text{if } Z_i + y_{ic} > y_3 \tag{9a}$$

$$Z_i + y_i = y_3 \qquad \text{otherwise} \tag{9b}$$

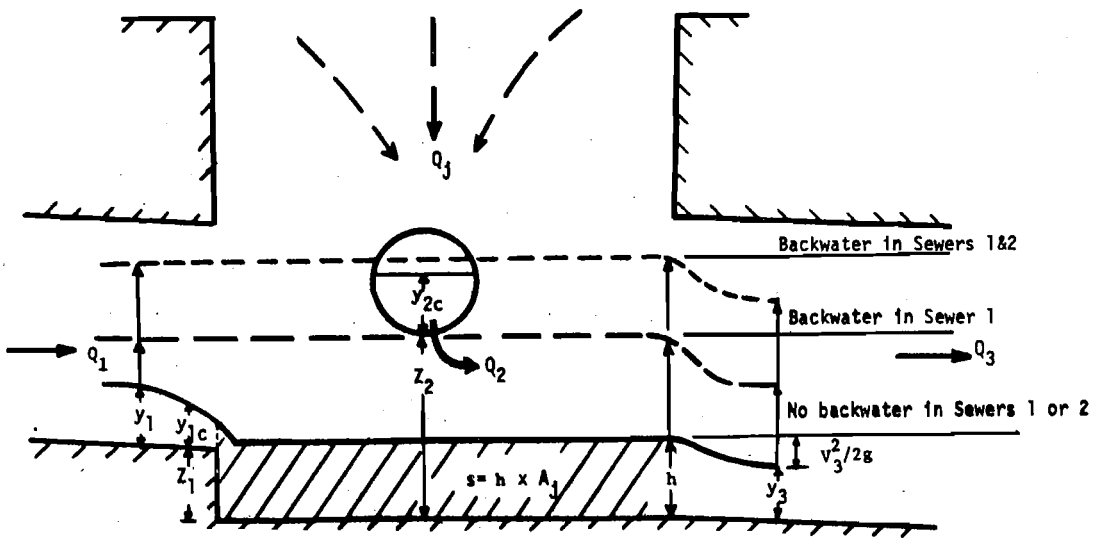
in which y_{ic} is the critical flow depth corresponding to the instantaneous flow rate Q_i, and Z_i is the height of the drop at the exit of inflowing sewers. Flow in the outflow sewer may be subcritical or supercritical. In the latter case y_3 in Eq. 9 is equal to the critical flow depth, y_{3c}, corresponding to the instantaneous flow rate Q_3.

For supercritical flow in the inflowing sewers i = 1 or 2, the flow is assumed to discharge freely into the junction. Thus, the discharge is computed independently without considering the flow condition in the junction, i.e., without specified downstream boundary condition. If both sewers 1 and 2 have supercritical flow, the discharge in the outflow sewer is then computed directly by using Eq. 8.

Although three-way junctions are most common in sewer networks, there also exist junctions with two, four or more joining sewers. Two-way junctions, which are located where there is a significant change in dimension, slope or horizontal alignment along the course of a sewer, can be considered as a special case of the three-way junction by letting i = 1 and 3 to represent the inflowing and outflowing sewers, respectively. Consequently, Eq. 8 with $Q_2 = 0$ and Eq. 9 with i = 1 can again be used.



a. Point-type junction



b. Reservoir-type junction

FIG. 2. SCHEMATIC OF SEWER JUNCTIONS

At a four-way junction, on the other hand, the flow in one of the three inflowing sewers is assumed to be free from the backwater effects of the junction. Consequently the inflow from this particular sewer is computed independently and treated as direct inflow (e.g., $Q_j(t)$ in Fig. 2) to the junction. Such a simplification appeared to be necessary for the sake of efficiency of the ISS Model. As can be seen from Fig. 2, for a three-way junction there already exits 18 possible combinations of flow conditions each of which requires the simultaneous solution of different sets of six nonlinear algebraic equations in six unknowns, depending on whether the flows are subcritical, critical, or supercritical. At a four-way junction without the aforementioned simplification, there would be 54 possible combinations of flow conditions each of which requires the simultaneous solution of a different set of eight nonlinear algebraic equations in 8 unknowns, making the programming of the computational algorithm extremely difficult.

(B) Reservoir-Type Junction.

The reservoir-type junction has a relatively large storage capacity and behaves like a reservoir, with a practically horizontal water surface which may change with time. The net discharge into the junction is equal to the time rate of change of storage in the junction, i.e.,

$$Q_1 + Q_2 + Q_j - Q_3 = \frac{ds}{dt} = A_j \frac{dh}{dt} \tag{10}$$

in which A_j is the constant horizontal cross-sectional area of the junction box and h is the water depth in the junction (Fig. 2b). Since the junction behaves like a reservoir, h can be assumed equal to the specific energy of the flow at the entrance of the outflowing sewer. Hence,

$$h = y_3 + \frac{V_3^2}{2g} \tag{11}$$

Furthermore, the junction is assumed capable to absorb and dissipate all the kinetic energy of the inflows. Thus, for subcritical flow in the inflow sewers, i = 1 and 2 (Fig. 2b),

$$Z_i + y_i = h \quad \text{if } Z_i + y_{ic} < h \tag{12a}$$

$$y_i = y_{ic} \quad \text{otherwise} \tag{12b}$$

If the flow in the outflowing Sewer 3 is supercritical, critical flow condition exits at its entrance and hence h in Eq. 12 is equal to the minimum specific energy h_c corresponding to the instantaneous flow rate Q_3. Supercritical flow in Sewers 1 and/or 2 is once again assumed to be discharging freely into the reservoir, and hence they are computed independently without considering the instantaneous flow conditions in the junction or at the entrance of Sewer 3.

Similar to two- and four-way point-type junctions, two-way reservoir junctions are treated as a special case of three-way junction, and for a junction with more than three sewers, only three of them are treated with direct backwater effects from the junction; additional sewers are merely regarded as direct inflow or outflow. A two-way reservoir junction, besides representing an actual storage element with or without a direct inflow or outflow, may approximately represent a concentrated energy loss along the course of a sewer.

2.2. Method of Solution

For a tree-type network consisting of N sewers, the St. Venant equations together with two initial and two boundary conditions for each of the sewers represent N distinct initial-boundary value problems. In the ISS Model numerical solution of these problems is obtained by using the

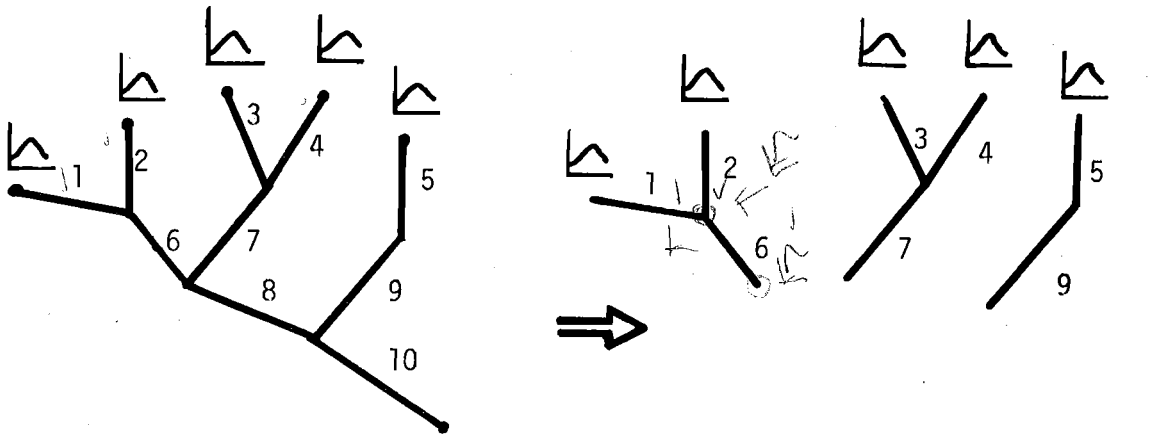
method of overlapping segments which is described as follows:

a) Numerical solutions are first obtained over those Y-segments whose inflowing sewers are connected to the inlet catch basins. The prescribed inflow hydrographs at the inlets, the compatibility conditions at the junction, and the known flow condition at the downstream end of the segment (or alternatively, if the downstream condition is unknown, the forward differences as a substitute) are used as the boundary conditions. For instance, for the network shown in Fig. 3a, the solutions are first obtained over the three Y-segments shown in Fig. 3b.

b) The use of forward differences as a substitute for the unknown boundary condition at the downstream end of a Y-segment is assumed to affect only the solutions obtained over the outflowing sewer of the segment. After the computation for the Y-segment is completed*, this first trial solution for the outflowing sewer is discarded but the "true" solution for the inflowing sewers of the segment is retained. Thus the inflow hydrographs into the junction of the current Y-segments are obtained. For instance, the solutions for the inflowing Sewers 1 through 5 of the three Y-segments shown in Fig. 3b are retained, and hence the inflow hydrographs to the junctions upstream of Sewers 6, 7 and 9 are obtained.

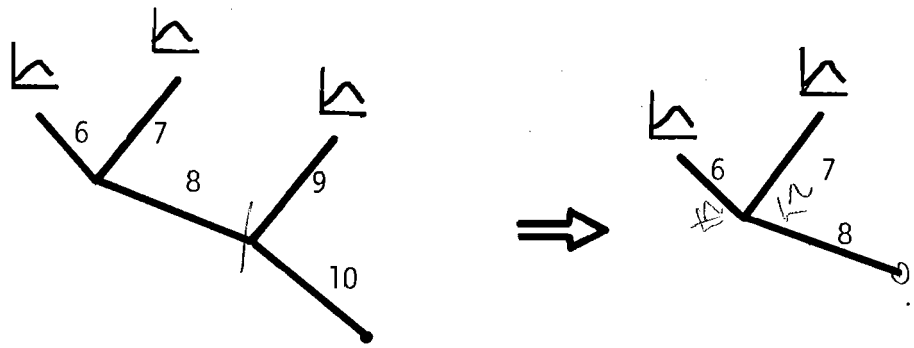
c) The inflow sewers of the current Y-segments are trimmed in the computations. The junction of the current Y-segments are then treated as the inlets of the advanced segments of the pruned network. Steps (a) and (b) just described are then applied to the new segments. This procedure is repeated in sequence over the network until it is reduced to the last Y-segment at the outlet of the network. For instance, for the network shown in Fig. 3a,

*The computations over a segment is terminated when the computed inflow into the outflowing sewer recedes and drops to 1.05 of the initial baseflow magnitude.



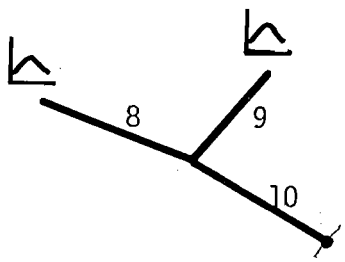
a. Complete solution domain

b. Solution domains for first-order sewers (1 through 5)



c. Reduced solution domain

d. Solution domain for second-order sewers (6 and 7)



e. Final solution domain for the sewers 8, 9, and 10

FIG. 3. SCHEMATIC OF SOLUTION BY METHOD OF OVERLAPPING SEGMENTS

Sewers 1 through 5 are trimmed, and the junctions at the upstream of Sewers 6, 7 and 9 are treated as the inlets of the pruned network shown in Fig. 3b. The numerical solution is then obtained for the new Y-segment covering Sewers 6, 7 and 8. The solution domain is now reduced to the last Y-segment as shown in Fig. 3e.

d) For the last segment, the prescribed boundary condition at its downstream end (outlet of the network) is used and thus the numerical solution over the entire network is completed.

2.3. Initial Conditions

In solving the St. Venant equations numerically, it is necessary to know all the depths and velocities along the sewer network at a given initial time. For combined sewers, these initial depths and velocities can be estimated from dry-weather flow conditions. For storm sewers, generally the only known initial condition is that of zero depth and velocity throughout the network. This dry-bed initial condition, however, constitutes a singularity and the numerical methods fail to advance the solution to the next immediate time level. Thus, it is assumed herein that initially a constant base flow, no matter how small and negligible from practical viewpoint, occurs along each sewer in the network.

The initial steady nonuniform flow profiles and velocities are computed by using the direct step method (1). Supercritical baseflow is free from backwater effect of the downstream junction and the flow control is the critical depth, y_c, at the entrance of the sewer.

For a subcritical baseflow the control is at the exit of the sewer. Thus, the depth of flow, y, at the sewer exit is given by Eq. 9 or Eq. 12 depending on whether the sewer is connected to a point-type or a reservoir-type junction (Fig. 2) at its downstream end. The overlapping segment scheme

described in the preceding section is then adapted together with the direct step method to solve for the initial flow profiles and velocities in the network.

For the last sewer of the network, the prescribed initial flow depth over the control structure at the outlet is used as the flow control in the computation of backwater profiles.

2.4. Procedure for Design of Sewer Sizes

In the ISS Model a built-in systematic design algorithm is provided for the determination of sewer sizes. For a given system layout, with specified slope for each sewer, the sewer sizes are computed as follows:

a) The diameters of the first order* sewers which emanate from inlet catch basins (e.g., Sewers 1 through 5 in the network shown in Fig. 3a) are initially estimated by using the Darcy-Weisbach equation for steady full pipe flow

$$D = 0.48 \left(f \frac{Q_p^2}{S_o}\right)^{1/5} \tag{13}$$

in which D is the diameter, Q_p is the peak discharge of the prescribed design inflow hydrograph at the upstream end of the sewer, S_o is the bottom slope of the sewer, and f is the Weisbach resistance coefficient. The computed diameter is then replaced by the nearest smaller commercially available pipe diameter.

---

* The sewer ordering system adopted here is the same as that proposed by Strahler for streams (3).

b) The design inflow hydrographs at the inlets are routed through the first-order sewers using a linear kinematic-wave approximation by assuming a constant propagation velocity, V_o, equal to that for the half-full, steady, uniform flow; i.e.,

$$Q(L, t) = Q(0, t-\tau) \tag{14}$$

in which L is the sewer length, and the time lag $\tau = L/V_o$. The inflows entering into the junction at the downstream end of the first order sewers are thus computed. From the sum of these inflows together with the direct junction inflow, $Q_j(t)$, if any, an estimate of the peak inflow into the outflowing sewer from the junction is obtained.

c) By using the estimated peak inflows into the sewers outflowing from the junction (e.g., Sewers 6, 7 and 9 of the network shown in Fig. 3a), first approximation for their diameters are calculated as described in Step (a).

d) With the sewer sizes so determined, the known inflow hydrographs are routed through the Y-segment by the numerical solution of the St. Venant equations. If the maximum depth of flow in the inflowing sewers of the Y-segment does not fall within the range between 0.8D and 1.0D, or the flow in the outflowing sewer becomes pressurized, the next larger or smaller commercially available pipe diameter is selected, and the numerical solution in this step is repeated. If the maximum flow depth in the inflowing sewers cannot be restricted within 0.8D to 1.0D because of the discrete sizes of the commercially available pipes, then the available commercial size which gives a depth nearest but smaller than 0.8D is used. This procedure (Steps (a) through (d)) is repeated for all other upstream segments (e.g., the three Y-segments shown in Fig. 3b).

e) With the sewer diameters and junction inflow hydrographs computed for all the upstream Y-segments, the method of overlapping segments is applied. The inflow sewers of the upstream Y-segments are trimmed. The junctions of these segments are treated as the inlets of the pruned network. The outflowing sewers of the previous upstream segments for which the diameters have been computed approximately in Step (c) become the inflowing sewers of the new Y-segment. The outflow hydrographs from the junctions of the previous Y-segments become the inflow hydrographs for the new segment.

f) Steps (b) through (e) are repeated until the diameters of all the sewers except those for the last Y-segment are computed.

g) The network is now reduced to the last Y-segment as shown in Fig. 3e. The diameters of the sewers in this last segment are then determined by exactly the same procedure, Steps (b) through (d), except that in Step (d) the downstream boundary condition (at the outlet of the network) is specified.

The above design algorithm in its current implementation occaionally gives a result consisting of nearly-full low-(first-) order sewers joining to a higher (second-) order sewer which is flowing at below capacity with a depth of 0.7D or smaller. This condition is a consequence of the effort to avoid pressurized flow in the sewer system. It occurs when a short first-order sewer is under significant backwater influence from the following long second-order sewer. Since in each cycle of the iteration, the sewer diameters are computed first for the lower order sewers, then for the higher order sewer, it may increase the diameter of the latter sewer rather than the former to satisfy the requirement of free surface flow. Since large sewers are more expensive, from the cost viewpoint this condition is undesirable. An algorithm, of course, can be implemented to correct this situation. However, as in its present form, inclusion of such an

algorithm in the Model would reduce considerably its efficiency. Should such a condition occur, the user can modify a few of the computed diameters after examining the predicted flow conditions in the network system designed. The ISS Model can then be rerun to simulate the flow conditions in the modified system in order to ensure that the design criteria are satisfied.

2.5. Limitations of ISS Model

The ISS Model is applicable to tree-type sewer systems which are characterized by (a) a set of inlet catch basins from each of which only one sewer emanates, (b) a set of junctions at each of which two or three sewers meet, and (c) a single outlet (Fig. 1).

A sewer system may include regulating and operational devices, such as gates, valves, weirs, overflows, regulators and pumping stations. Some of these control structures provide flow controls dividing hydraulically a complex system into a number of subsystems. The ISS Model can be applied to these subsystems one at a time in an appropriate sequence and the results can then be integrated together to give the simulation over the entire sewer system.

The ISS Model, in its current implementation, can simulate the flow with regulating and operational devices if they are located at the system outlet. From a hydraulics viewpoint, the control facilities which can be handled by the ISS Model are characteristically described by one of the following four types of flow equations:

a) stage-time relationship

$$y = f_1(t) \tag{15}$$

b) velocity-depth relationship

$$V = f_2(y) \tag{16}$$

c) discharge-depth relationship

$$Q = f_3(y) \tag{17}$$

d) discharge-time relationship

$$Q = f_4(t) \tag{18}$$

The first type of control, Eq. 15, exists if the sewer system discharges into a large river, a lake, or an ocean, with a known surface variation. The second and third types of control, Eqs. 16 and 17, respectively, occur if a weir, a gate, or a similar flow regulating device exists at the outlet. The last type of control, Eq. 18, exists if the outflow from the sewer system is pumped or released at a prescribed rate to a receiving water body.

As to the physical characteristics of sewers and junctions, the following restrictions exist for the Model:

a) The sewers are circular in cross section.

b) The inflow of storm water into the sewer system occurs only at discrete nodal points, viz., inlets, junctions and manholes.

c) Manholes and junction boxes are open to atmospheric pressure. At both the manholes and junctions, invert lines of the upstream incoming sewers are at the same or high elevation than that of the downstream outflowing sewer.

d) For a junction or manholes with more than three joining sewers, only three can be considered for direct backwater effects. Others (preferably those sewers with small backwater effect from the junction) are treated as direct inflow (e.g., Q_j in Fig. 2.).

In regard to the limitations on flow conditions, the ISS Model cannot simulate the occurrence of pressurized conduit flows. Neither can

it account for moving hydraulic jumps and surges within the sewers. Furthermore, as stated in Sec. 2.3, the Model cannot handle dry-bed initial condition because of the computational singularity at such condition. Discussion on the significance of these limitations and possible remedies has been reported elsewhere (2, 5).

3. NUMERICAL SOLUTION TECHNIQUE

3.1. Difference Equations

In the ISS Model, numerical solutions are attempted on the characteristic form of the St. Venant equations (2, 4)

$$dV + (gB/A)^{1/2}\, dy + g(S_f - S_o)\, dt = 0 \qquad (19a)$$

$$dx = [V + (gA/B)^{1/2}]\, dt \qquad (19b)$$

C^+

$$dV - (gB/A)^{1/2}\, dy + g(S_f - S_o)\, dt = 0 \qquad (20a)$$

$$dx = [V - (gA/B)^{1/2}]\, dt \qquad (20b)$$

C^-

in which the notation C^+ and C^- indicate the forward and backward characteristics, respectively. Equations 19 and 20 are solved by using a first-order finite difference scheme, on a fixed rectangular grid (Fig 4). Thus,

$$x_P - x_R = [V_R + (gA_R/B_R)^{1/2}]\Delta t \qquad (21a)$$

$$V_P - V_R + (gB_R/A_R)^{1/2}(y_P - y_R) + g(S_{fR} - S_o)\Delta t = 0 \qquad (21b)$$

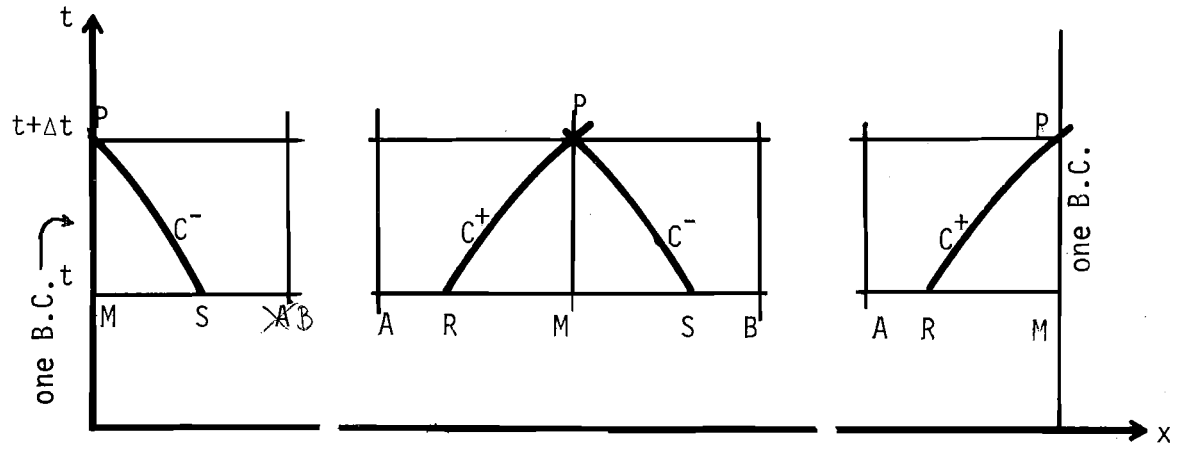
C^+

$$x_P - x_S = [V_S - (gA_S/B_S)^{1/2}]\Delta t \qquad (22a)$$

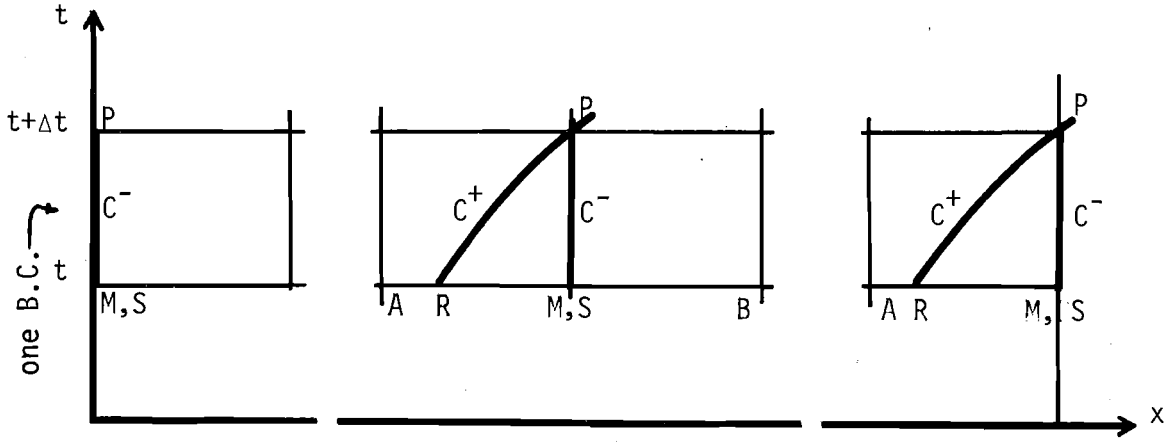
$$V_P - V_S - (gB_S/A_S)^{1/2}(y_P - y_S) + g(S_{fS} - S_o)\Delta t = 0 \qquad (22b)$$

C^-

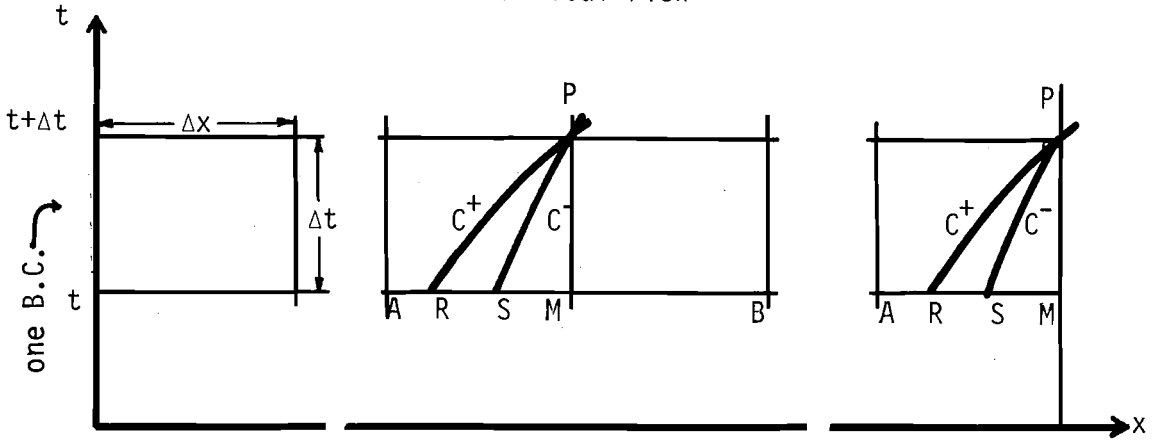
in which the subscripts R and S refer to the computational grid points at time t (Fig. 4), where the solution is known at regular distance increments (grid points A, M, and B) either as given initial conditions or as the results from previous computations; and the subscript P refers to the grid point at



a. Subcritical Flow



b. Critical Flow



c. Supercritical Flow

FIG. 4. RECTANGULAR COMPUTATION GRID FOR METHOD OF CHARACTERISTICS

the time $t + \Delta t$ where the solution is sought. The time increment Δt is selected so as to confirm with the Courant criterion for stability,

$$\Delta t \leq \frac{\Delta x}{V + \sqrt{gA/B}} \tag{23}$$

in which Δx is the selected distance increment.

By applying a linear interpolation for velocity V between points A and M (Fig. 4), V_R can be expressed as

$$V_R = \frac{V_M - [\theta(V_M - V_A)(gA_R/B_R)^{1/2}]}{1 + \theta(V_M - V_A)} \tag{24}$$

in which $\theta = \Delta t/\Delta x$. Likewise, linear interpolation for depth y between A and M, with the aid of Eq. 21a, yields

$$y_M - y_R - \frac{\theta(y_M - y_A)}{1 + \theta(V_M - V_A)} [V_M + (gA_R/B_R)^{1/2}] = 0 \tag{25}$$

In the computation, Eq. 25 is first solved for y_R using the Newton-Raphson method, and then V_R is obtained from Eq. 24 explicitly.

The values of y_S and V_S are similarly found by applying a linear interpolation between points M and B

$$y_S - y_M + \frac{\theta(y_B - y_M)}{1 - \theta(V_B - V_M)} [V_M - (gA_S/B_S)^{1/2}] = 0 \tag{26}$$

$$V_S = \frac{V_M - [\theta(V_M - V_B)(gA_S/B_S)^{1/2}]}{1 - \theta(V_M - V_B)} \tag{27}$$

for subcritical flow; and by applying a linear interpolation between points A and M

$$y_S - y_M + \frac{\theta(y_S - y_A)}{1 + \theta(V_M - V_A)} [V_M - (gA_S/B_S)^{1/2}] = 0 \tag{28}$$

$$V_S = \frac{V_M + (V_M - V_A)\theta(gA_S/B_S)^{1/2}}{1 + \theta(V_M - V_A)} \tag{29}$$

for supercritical flow.

With the values V_R, y_R, V_S and y_S computed, Eqs. 21 and 22 can now be reduced to a simpler form of two linear simultaneous equations in two unknowns, y_P and V_P,

$$V_P = W_1 - F_R y_P \tag{30a}$$

$$V_P = W_2 + F_S y_P \tag{30b}$$

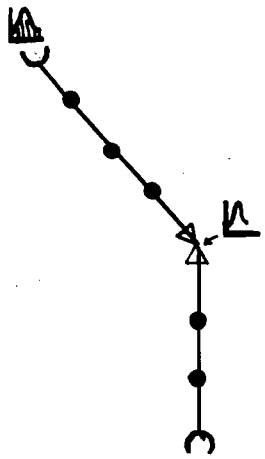
in which

$$W_1 = V_R + F_R y_R - g(S_{fR} - S_o)\,\Delta t \tag{31a}$$

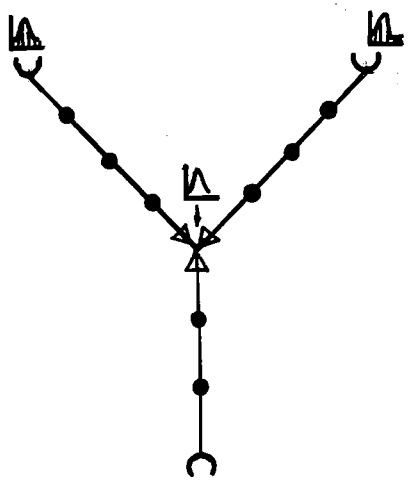
$$W_2 = V_S - F_S y_S - g(S_{fS} - S_o)\,\Delta t \tag{31b}$$

and $F = (gB/A)^{1/2}$

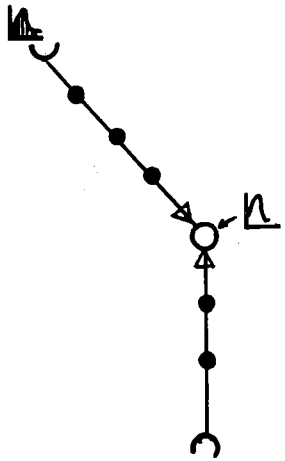
In the remainder of this chapter, y_R, V_R, y_S, and V_S will be assumed to be obtained from the solution of Eqs. 24 through 29 in the previous stage of the calculation. Therefore, simplified form of the forward and backward characteristics, Eqs. 30a and 30b, will be used in the computations at all the stations. To identify various possible stations, a schematic description of a computational segment is shown in Fig. 5.



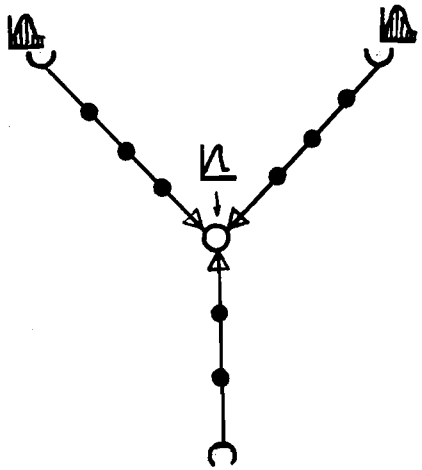
a. Two sewers joining at a point junction



b. Three sewers joining at a point junction



c. Two sewers joining at a reservoir juction



d. Three sewers joining at a reservoir junction

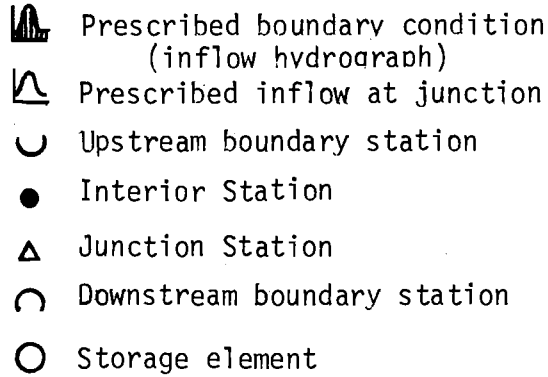


FIG. 5. OVERLAPPING SEGMENTS

3.2. Interior Stations

An interior station is a grid point within the computational segment where at least one other grid point exists both upstream and downstream within the same reach (Fig. 5). At such a station both the forward and backward characteristic equations apply (Fig. 4). Thus Eqs. 30a and 30b can be solved simultaneously for y_P and V_P:

$$y_P = \frac{(W_1 - W_2)}{(F_R + F_S)} \tag{32}$$

$$V_P = \frac{(F_S W_1 + F_R W_2)}{(F_R + F_S)} \tag{33}$$

If the flow at the upstream end of the sewer is critical, the depth and velocity at the first interior station are computed by using the second-order Lax-Wendroff scheme as follows:

$$y_P = y_M - 0.5\,\theta\left[\frac{A_M}{B_M}(V_B - V_A) + V_M(y_B - y_A)\right] + 0.5\,\theta^2\left(g\frac{A_M}{B_M} + V_M^2\right)(y_B - 2y_M + y_A)$$

$$+ \theta^2\left[\frac{A_M}{B_M} V_M(V_B - 2V_M - V_A)\right] \tag{34}$$

and

$$V_P = V_M - 0.5\,\theta\left[V_M(V_B - V_A) + g(y_B - y_A) + 2g\Delta x(S_{fM} - S_o)\right]$$

$$+ 0.5\,\theta^2\left[V_M^2 + \frac{gA_M}{B_M}\right](V_B - 2V_M + V_A) + \theta^2 g V_M(y_B - 2y_M + y_A) \tag{35}$$

In order to save computer time, computations at an interior station do not start until the arrival of either the downstream-moving input flood wave or the upstream-moving backwater wave. Arrival of these waves at the station is determined by an inspection of discharges at the immediate upstream and downstream stations. If the discharges at these neighboring stations are found to be different from that of the baseflow, the computations are performed, otherwise the computations are bypassed, and the values of y and V corresponding to the initial baseflow conditions are retained.

3.3. Upstream Boundary Station

At an upstream boundary station of a computational segment (Fig. 5), the inflow hydrograph is known, that is,

$$Q(0,t) = q \tag{36}$$

in which $q = q(t)$ is a known function of time either in the form of specified inflow hydrographs at the inlets (for those segments which include first-order sewers) or in the form of previous solutions of the adjacent upstream segments. Thus, at any instant, the velocity of flow at an upstream boundary station can be expressed as

$$V_P = \frac{q}{A_P} \tag{37}$$

in which the cross-sectional area of the flow, A_P, is a function of the depth, y_P.

For a subcritical flow, the backward characteristic (Eq. 31b) involving points M, A, and P (Fig. 4) is applicable at the upstream boundary. Thus, substituting Eq. 37 into Eq. 31b,

$$\frac{q}{A_P} - F_S y_P - W_2 = 0 \tag{38}$$

This equation is a nonlinear function of the unknown depth y_P and it is solved

by iteration using the Newton-Raphson method. The value of V_P is then computed from Eq. 37 explicitly.

For a supercritical flow, the critical flow condition at the entrance

$$V_P = (gA_P/B_P)^{1/2} \tag{39}$$

together with Eq. 37 is used to obtain the solutions at the upstream boundary station. Elimination of V_P from Eqs. 37 and 39 yields

$$\frac{q}{A_P} = (gA_P/B_P)^{1/2} \tag{40}$$

Since A_P and B_P are known functions of y_P this equation is solved for y_P using the Newton-Raphson method, and then the value of V_P is computed explicitly from Eq. 37.

3.4. Downstream Boundary Station

As shown in Fig. 4, for a supercritical flow both the forward and backward characteristics, Eqs. 30a and 30b, are applicable at the downstream boundary station of the segment (Fig. 5). These two equations are solved simultaneously as has been done for the interior stations. Subsequently, the unknowns y_P and V_P are computed explicitly using Eqs. 32 and 33, respectively.

If a free-fall exists at the downstream boundary station, the continuity equation, Eq. 1, is solved directly by using forward differences between points A and M (2). Thus, linear approximation gives

$$\frac{\partial y}{\partial x} = \frac{y_M - y_A}{\Delta x} \tag{41}$$

$$\frac{\partial V}{\partial x} = \frac{V_M - V_A}{\Delta x} \tag{42}$$

and

$$\frac{\partial y}{\partial t} = \frac{y_P - y_M}{\Delta t} \tag{43}$$

Substitution of Eqs. 41 to 43 into Eq. 1 yields the following explicit expression to compute the unknown flow depth, y_P,

$$y_P = y_M - 0.5\,\theta[(V_M + V_A)\,(y_M - y_A) + (\frac{A_M}{B_M} + \frac{A_A}{B_A})\,(V_M - V_A)] \tag{44}$$

With the value of y_P calculated, V_P is then obtained explicitly by using the critical flow condition (Eq. 39) at the downstream boundary station.

For a subcritical flow at the downstream boundary station, the forward equation (Eq. 30a) together with a prescribed boundary condition is used for the computation of flow depth y_P and velocity V_P. As it has been stated in Sec. 2.2, for all the segments except the last one at the outlet of the network or at definite control sections, this downstream boundary condition is not known. Therefore, the continuity and momentum equations (Eqs. 1 and 2) are solved by using forward differences between points A and M (Fig. 4) at the downstream boundary. The value of y_P is computed from Eq. 44, and then V_P is obtained explicitly from the finite difference form of the momentum equation (Eq. 2),

$$V_P = V_M - \theta[V_M(V_M - V_A) + g(y_M - y_A)] + g\Delta t(S_o - S_{fM}) \tag{45}$$

At the outlet of the network (downstream boundary station of the root segment) a number of possible boundary conditions may be specified depending on the type of control existing there. One of them is the free-fall (critical flow) condition which has already been described in this section. Other common types of control, which have been described in Sec. 2.4,

include the stage-time relationship (Eq. 15), velocity-depth relationship (Eq. 16), discharge-depth relationship (Eq. 17) and discharge-time relationship (Eq. 18). In the case of a stage-time relationship, the value of y_p is computed directly from Eq. 15, and then V_p is obtained explicitly from Eq. 45. For the remaining types of control, the value of y_p is first computed by using Eq. 44, and then the value of V_p is obtained from the prescribed control equation, i.e., Eqs. 16, 17, or 18.

3.5. Junction Stations

As discussed in Subsection 2.1.2, two types of junctions, depending on the relative size of the junction box, are considered in the ISS Model. They are the point-type and the reservoir-type junctions, and they can be further classified here as three-way and two-way junctions depending on the number of joining sewers (Fig. 5).

3.5.1. Three-Way Point Junction

A computational grid in the vicinity of a three-way point junction is shown in Fig. 6. The solution is sought at the grid points P^I, P^{II} and P^{III}, which represent the junction stations of Sewers I, II, and III, respectively. Six equations are required for the solution of the six unknowns; namely, the depth and velocity at each of the three stations, P^I, P^{II} and P^{III}. The equation of continuity at the junction and the forward characteristic equations at the stations P^I and P^{II} furnish three of the needed equations:

$$Q_j + V_P^I A_P^I + V_P^{II} A_P^{II} - V_P^{III} A_P^{III} = 0 \tag{46}$$

$$V_P^I = W_1^I - F_R^I y_P^I \tag{47}$$

$$V_P^{II} = W_1^{II} - F_R^{II} y_P^{II} \tag{48}$$

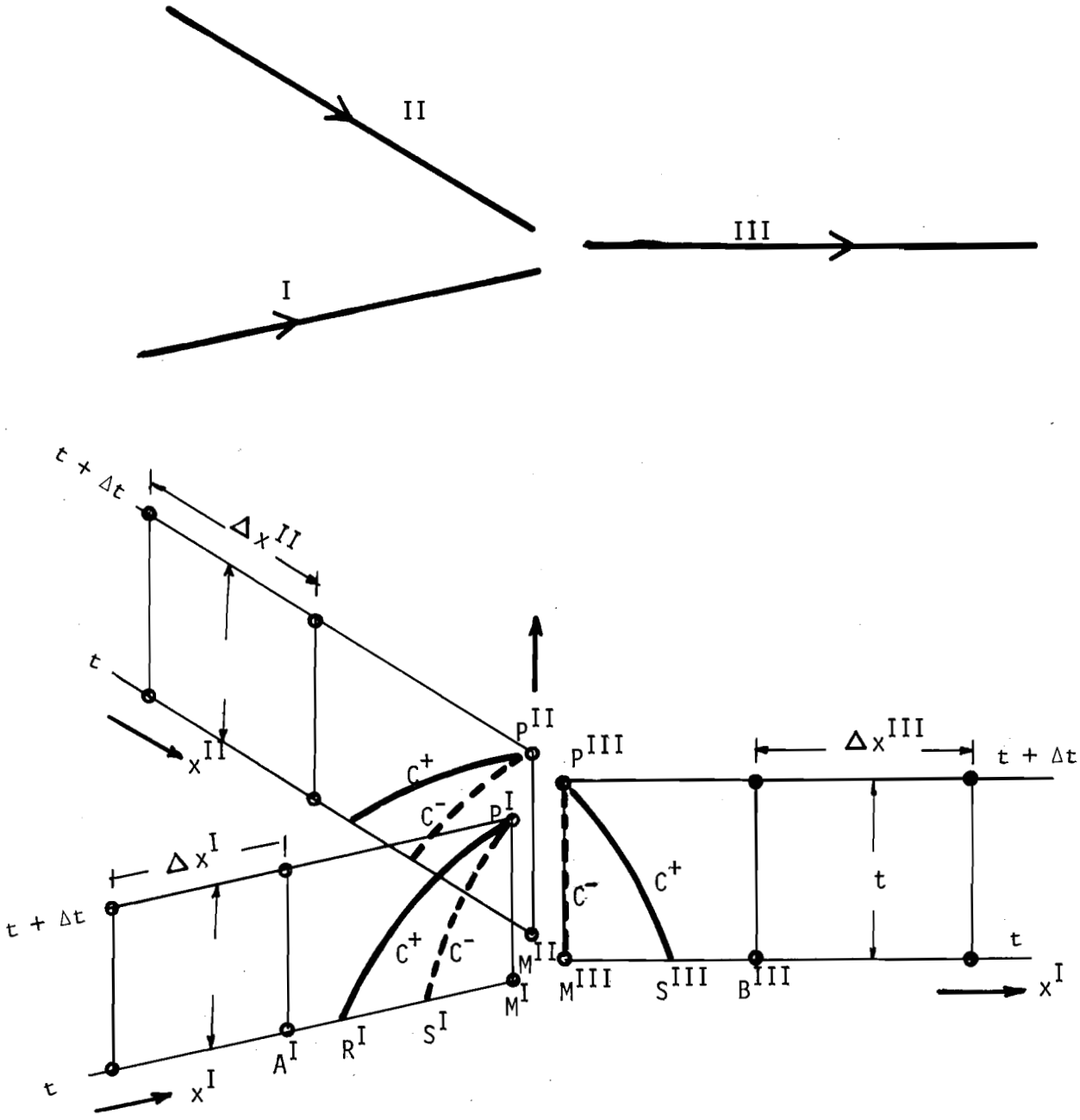


FIG. 6. CHARACTERISTICS AT JUNCTION

The remaining three equations, one for each station, are provided either by the backward characteristic equation, or by the critical flow condition or the kinematic compatibility condition, depending on the flow regime. Thus,

a) at Station P^{I}

$$V_P^I = W_2^I + F_S^I \, y_P^I \tag{49a}$$

for supercritical flow;

$$V_P^I = (gA_P^I/B_P^I)^{1/2} \tag{49b}$$

for critical flow; and

$$Z_I + y_P^I = y_P^{III} \tag{49c}$$

for subcritical flow;

b) at Station P^{II}

$$V_P^{II} = W_2^{II} + F_S^{II} \, y_P^{II} \tag{50a}$$

for supercritical flow;

$$V_P^{II} = (gA_P^{II}/B_P^{II})^{1/2} \tag{50b}$$

for critical flow; and

$$Z_{II} + y_P^{II} = y_P^{III} \tag{50c}$$

for subcritical flow; and

c) at Station P^{III}

$$V_P^{III} = (gA_P^{III}/B_P^{III})^{1/2} \tag{51a}$$

for critical flow; and

$$V_P^{III} = W_2^{III} + F_S^{III} \, y_P^{III} \tag{51b}$$

for subcritical flow.

Equations 49, 50, and 51 provide 18 different possible combinations of group of three equations, one from each of the three stations P^I, P^{II}, and P^{III}, depending on whether the flows at these stations are subcritical, critical, or supercritical. The appropriate group of three equations to be used in conjunction with Eqs. 46, 47, and 48 to advance the numerical solution from the time t to t + Δt (Fig. 6) is determined by an inspection of the Froude number at the grid points M^I, M^{II}, and M^{III}.

From a computational viewpoint, the 18 possible sets of six equations to be solved for the six unknowns can be classified into three major categories. In the first category are eight equation sets from which the unknowns can be obtained by simultaneous solution of two equations at each of the junction stations. This is the case if neither of the flows at Station P^I and P^{II} is subcritical, and hence the solutions can be obtained from Eqs. 47 and 49a or b at P^I, from Eqs. 48 and 50a or b at P^{II}, and then from Eqs. 46 and 51a or b at P^{III}. Computational aspects of these solutions have been described in the preceding sections in relation to upstream and downstream boundary stations.

In the second category are the eight equation sets which require the solution of two and then four equations simultaneously for the evaluation of the six unknowns. This is the case if the flow at one of the stations P^I or P^{II} is subcritical. Assuming, for instance, the flow at Station P^I to be subcritical, Eqs. 48 and 50a or b are first solved for V_P^{II} and y_P^{II}, and then Eqs. 46, 47, 49c and 51a or b are solved simultaneously for y_P^I, V_P^I, y_P^{III} and V_P^{III}. In this latter solution, Eqs. 47, 49c and 51a or b are first substituted

into Eq. 46 to yield a nonlinear equation, $f_5(y_P^{III}) = 0$. This equation is solved for y_P^{III} by iteration using the Newton-Raphson method, and then values of y_P^I, V_P^I, and V_P^{III} are obtained from Eqs. 49c, 47, and 51a or b, respectively.

The third category includes two equation sets from which the six unknowns can be evaluated only by simultaneous solution of all the six equations. This is the case if the flows at both of the stations P^I and P^{II} are subcritical, and hence Eqs. 46, 47, 48, 49c, 50c, and 51a or b are coupled. In obtaining the solutions, Eqs. 47, 48, 49c, 50c, and 51a or b are substituted into Eq. 46 to yield a nonlinear equation, $f_6(y_P^{III}) = 0$. This equation is solved for y_P^{III} by iteration using the Newton-Raphson method, and then the values of y_P^I, V_P^I, y_P^{II}, V_P^{II} and V_P^{III} are obtained respectively from Eqs. 49c, 47, 50c, 48, and 51a or b.

3.5.2. Two-Way Point Junction

At a two-way point junction, such as the junction of Sewers I and III in Fig. 4a, four equations are required for the solution of the four unknowns: the depth and velocity at the two stations P^I and P^{III}. By eliminating all the terms having the superscript II in Eq. 46, the continuity equation is

$$Q_j + V_P^I A_P^I - V_P^{III} A_P^{III} = 0 \qquad (52)$$

Equation 52 together with Eqs. 47, 49, and 51 provides the required four equations.

From a computational viewpoint, two different cases can be considered in obtaining the solution for this set of four equations. The first case is that of subcritical flow at Station P^I, for which Eqs. 47, 49c, 51, and 52 are solved simultaneously for the unknowns y_P^I, V_P^I, y_P^{III} and V_P^{III}. The

second case is that of supercritical or critical flow at Station P^I, for which Eqs. 47 and 49a or b are solved simultaneously for y_P^I and V_P^I, and then Eqs. 51 and 52 are solved simultaneously for y_P^{III} and V_P^{III}.

3.5.3. Three-Way Reservoir Junction

Computations for the depth and velocity at the three stations of a reservoir junction proceed in much the same manner as that for a three-way point junction. For the only changes are the compatibility conditions, i.e., the equation of continuity and the kinematic compatibility conditions. The continuity equation to replace Eq. 46 is

$$Q_j + V_P^I A_P^I + V_P^{II} A_P^{II} - V_P^{III} A_P^{III} = \frac{A_j}{\Delta t} \left(y_P^{III} + \frac{{V_P^{III}}^2}{2g} - y_M^{III} - \frac{{V_M^{III}}^2}{2g}\right) \tag{53}$$

The energy compatibility conditions for the reservoir junction corresponding to Eqs. 49c and 50c are

$$Z^I + y_P^I = y_P^{III} + \frac{{V_P^{III}}^2}{2g} \tag{54}$$

$$Z^{II} + y_P^{II} = y_P^{III} + \frac{{V_P^{III}}^2}{2g} \tag{55}$$

For the case when neither of the flows at Stations P^I and P^{II} is subcritical, the solution is obtained by the same procedure as described for the three-way point junction except that Eq. 46 is replaced by Eq. 53. In other words, y_P^I and V_P^I are obtained from the simultaneous solution of Eqs. 47 and 49a or b; y_P^{II} and V_P^{II} are obtained from the simultaneous solution of Eqs. 48 and 50a or b; and finally, y_P^{III} and V_P^{III} are computed from the simultaneous solution of Eqs, 51 and 53.

For the case when the flow is subcritical at either one of Stations P^I or P^{II}, say P^I, the solution is obtained by the following procedure:

a) Equations 48 and 50a or b are solved simultaneously for y_P^{II} and V_P^{II}, as described previously.

b) Substitution of Eqs. 47, 51 and 54 into Eq. 53 yields a nonlinear function, $f_7(y_P^I) = 0$. This equation is then solved numerically for y_P^I using the Newton-Raphson method.

c) From Eq. 47, the value of V_P^I can then be computed explicitly.

d) The value of y_P^{III} is obtained from Eqs. 54 and 51a or b through an interative solution using the Newton-Raphson method, and then V_P^{III} can be computed explicitly by using Eq. 51a or b.

For the case when the flows at both of the stations P^I and P^{II} are subcritical, the solution procedure is as follows:

a) Substitution of Eqs. 47, 48, 54, 55 and 51a or b into Eq. 53 yields a nonlinear function, $f_8(y_P^I) = 0$. This function is then solved numerically for y_P^I using the Newton-Raphson method.

b) From Eq. 47, V_P^I can then be computed explicitly.

c) From Eqs. 54 and 55, y_P^{II} is obtained explicitly. The result is then substituted into Eq. 48 to evaluate V_P^{II}.

d) The value of y_P^{III} is computed from Eqs. 54 and 51a or b through an iterative solution using the Newton-Raphson method, and then V_P^{III} can be computed explicitly from Eq. 51a.

3.5.4. Two-Way Reservoir Junction

At a two-way reservoir junction, such as the junction of Sewers I and III in Fig. 5c, the continuity equation is

$$Q_j + V_P^I A_P^I - V_P^{III} A_P^{III} = \frac{A_j}{\Delta t} \left(y_P^{III} + \frac{{V_P^{III}}^2}{2g} - y_M^{III} - \frac{{V_M^{III}}^2}{2g}\right) \qquad (56)$$

This equation, together with Eqs. 47, 51 and any one of Eqs. 49a, 49b, or 54, furnishes the required set of four equations.

In obtaining a solution for a two-way reservoir junction, if the flow at Station P^I is subcritical, Eqs. 47, 54, 56 and 51 are solved simultaneously for y_P^I, V_P^I, y_P^{III} and V_P^{III}. If the flow at Station P^I is supercritical (or critical), Eqs. 47 and 49a (or 49b) are solved simultaneously for y_P^I and V_P^I, and then Eqs. 51a or b and 56 are solved simultaneously for y_P^{III} and V_P^{III}.

4. IMPLEMENTATION OF ISS MODEL

4.1. General Information on Computer Program

The Illinois Storm Sewer System Simulation Model is programmed in PL/1 language. In this chapter an overall general view of the computer program together with requirements for computer capability and user's background knowledge is described. The program has been executed on the IBM System 360/75 of the University of Illinois at Urbana-Champaign, and it can easily be adapted for execution on most large IBM 360 or 370 models running under OS or VS operating systems.

The program can be implemented by using a distribution tape which will be described in Chapter 7 or by using the program listing given in Appendix A. The former approach is recommended, particularly for those users who are not technical oriented on computer operations. Programming experiences in PL/1 language will be helpful but not essential for the implementation of the program. In fact, through the use of the distribution tape, as described in Chapter 7, the user of the ISS Model is not even required to have previous experiences on FORTRAN or other computer languages. The material discussed in this chapter, together with those in Chapters 5 and 7, provides sufficient information for the execution of the ISS Model.

However, users who prefer to use the program listing (Appendix A) for implementation or modification should have adequate understanding of the following IBM Systems 360 and 370 concepts:

a) creation, use and modification of partitioned data sets for storage of source and object decks, as well as load modules;

b) use of the linkage editor to produce executable load modules, and method of executing the load modules thus created;

c) use of other IBM System utilities, such as IEHMOVE, IEBUPDTE, etc.;

d) use of tapes, disks, and other direct access devices; and

e) use of IBM job control language (commonly referred to as JCL).

Those who need to use the program listing and yet unfamiliar with computer applications are advised to seek assistance from experienced programmers.

The computer program of the ISS Model consists of around 3000 statements written in PL/1-source code, as well as an assembler language subroutine. The PL/1 portion includes several short external subroutines, a main controlling section with network control card input routines, and a large set of numerical computation routines. The program is written to be composed of modules, whenever possible, so that most of the routines are separately compiled and then linkage edited together with the assembler language routine to form the executable load module.

In implementation, the program resides on a direct-access storage device, such as a disk, a drum, or a data cell drive. This enables the program to be loaded quickly and efficiently into the main storage without entailing the added overhead of recompilation each time the program is executed.

As shown schematically in Fig. 7, the program begins its execution by reading a set of control and data cards which describes the run control specifications (user commands), sewer system layout and physical characteristics of system components and inflow hydrographs. After verifying the accuracy and completeness of this information, the program starts to execute the numerical simulation phase. Output from the numerical simulation consists of a tabular print out of the computed or specified sewer diameters, time variations of flow rate, velocity, and depth at sewer entrances, and space variations of flow rate, velocity, and depth along the sewers at specific time intervals. In addition to this default print out (the amount of which is under the control of user through various control card options), optional data capture facilities are provided which allow the user to produce his own

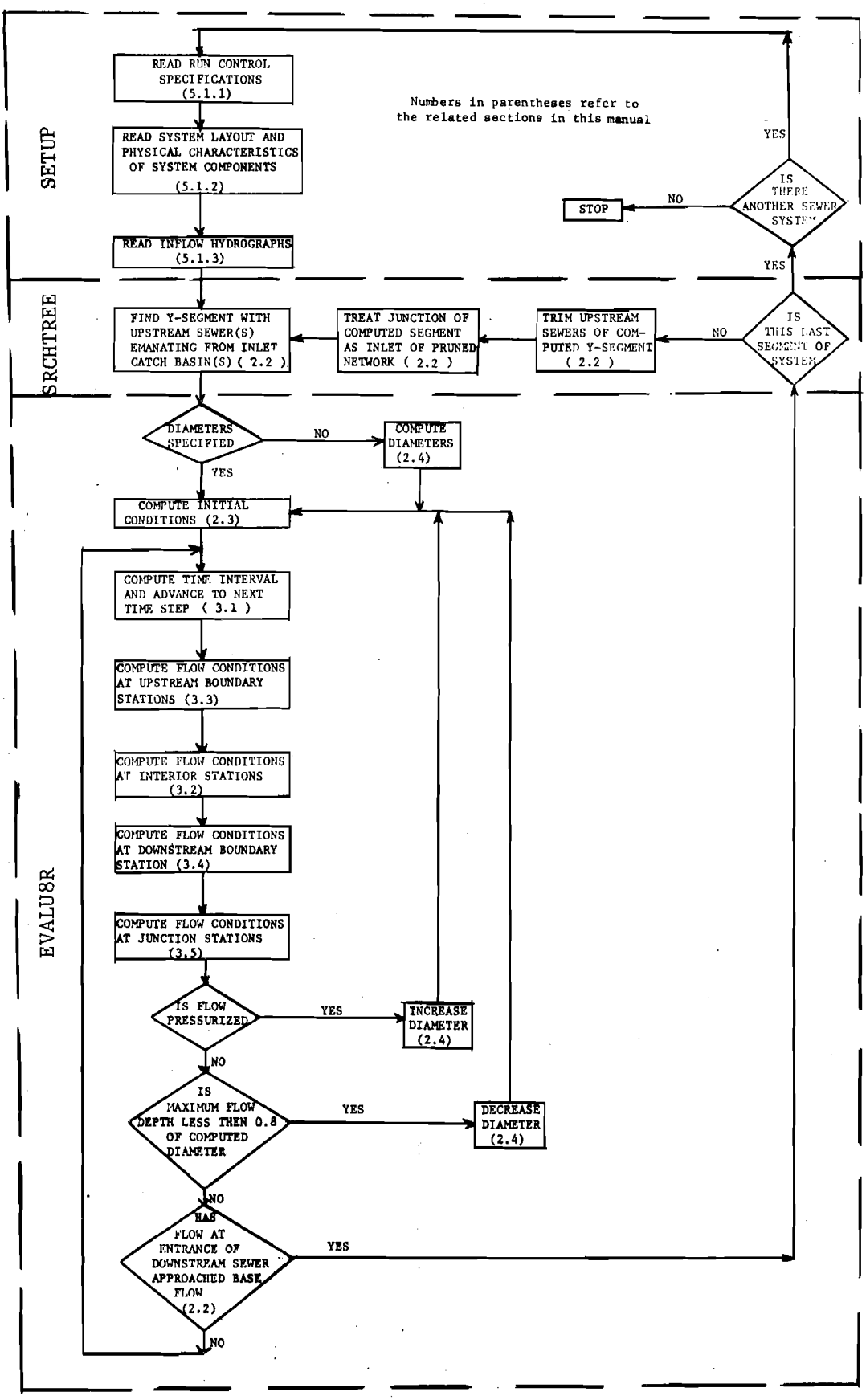


FIG. 7. MACRO FLOW CHART OF COMPUTER PROGRAM

plots or graphic displays of flow behavior at the inlets, junctions, manholes, and outlet of the network. After completion of the simulation for one sewer network, the program proceeds on to the next network, if any, until all such networks are processed.

4.2. Computer Requirements

The ISS Model is programmed to be operated on IBM System 360 or 370 with OS/360 (preferably OS/MVT) or its VS equivalent. The machine must have the floating point instruction set. It is believed possible to convert the program to run on a machine without the decimal instruction set, but this alternation is not recommended.

The program was originally written for the OS PL/1 (F) Compiler. But following the installation of the newer IBM OS PL/1 Optimizing Compiler (Program Product) on the University of Illinois IBM System 360/75, more recent portions of the program have been written using techniques to take advantage of this newer compiler. While the program can be compiled under either OS PL/1 (F) or OS PL/1 Optimizing Compiler, the latter is strongly recommended due to great reduction in block entry/exit overhead associated with allocation and freeing of automatically allocated storage.

The program will probably not be operated successfully under the IBM OS PL/1 Checkout Compiler without a significant amount of recoding. At any circumstance, the use of checkout compiler is not recommended due to its high overhead in contrast to the present program which involves significant amount of computations. For the rare cases where the IBM System 360 or 370 is not equipped with either OS PL/1 (F) or OS PL/1 Optimizing Compiler, the program can only be implemented through the use of a distribution tape. This, in fact, is another reason of preference of using the distribution tape to the program listing in implementation of the ISS Model.

Although it is recognized that PL/1 compilers are available for some other makes of computers, it would be difficult, if not impossible, to run the program in these computers without extensive reprogramming. This is due to the fact that several central concepts used throughout the program are based on the assumption of IBM System 360 or 370 implementation.

It is desirable to dedicate one demountable disk pack (2311, 2314, or 3330 type) for the purpose of maintaining the source and object decks, load modules, PL/1 Compiler (if necessary), and associated libraries in one central location. Therefore, the computer to be used should, if possible, have at least one spare disk drive for use by non-permanently mounted packs. If this technique is adopted, the disk pack may contain the following:

a) a partitioned data set of source programs (this data set is of a suitable format to be maintained with the IBM System utility program IEBUPDTE);

b) a partitioned data set of object decks;

c) a partitioned data set containing compilation and assembly listings;

d) a system catalog for data sets on the disk pack;

e) a data set to hold information generated by the PLOT option;

f) several data sets containing sample input decks and flow charts to aid in program documentation;

g) a general system library containing useful utility programs for maintaining the program; and

h) two libraries containing the PL/1 (F) Compiler and its libraries, in the event no other PL/1 compiler is available at the installation.

These data sets and libraries are automatically placed on the disk pack if the distribution tape is used.

4.3. Storage Requirements

The computer program of the ISS Model is currently running on the University of Illinois IBM System 360/75 in a region of 220K bytes of memory under OS/MVT and HASP. However, the requirement for the size of storage for a particular application may vary considerably depending on the following factors:

a) the size of the storm sewer network to be processed;

b) duration of the flood flow to be simulated;

c) the degree of discretization (i.e., magnitude of distance increments, Δx) used in the numerical solutions;

d) the use of GRAPH and PLOT options to be discussed in Subsection 5.1.1;

e) the type of PL/1 compiler used, and options specified for compilation and execution; e.g., the ISASIZE option (specified at execution time) for programs compiled by PL/1 optimizing compiler.

Storage requirements can be reduced, if necessary, by judicious use of overlay structures. However, in so doing caution must be exercised since a poor overlay structure could result in extreme overlaying inefficiencies. Another approach to reduce storage requirement would be to run the program under one of the VS operating systems available for most of IBM System 370 models. Even so, due to the program's modular structure, it is doubtful that the program could be run efficiently on real storage of the size much less than 100K bytes. Ideally, in a System 360 or 370, the user should try to have a region of around 300K available for use by the ISS Model, if possible.

5. INPUT AND OUTPUT OF COMPUTER PROGRAM

5.1. Input Data

The input into the computer program is user oriented. Each of the input data cards except those for time and discharge values of inflow hydrographs is identified by an alphameric tag. In general the input cards are free format, between the numbers commas and blanks can be used freely. The only format restriction is that the alphameric tag must begin in column 1, with at least one blank space separating the tag with the associated numerical data. Continuation cards are not permitted, since in this case one can simply punch a second card with the same alphameric tag. The input deck consisting of the control and data cards describes the following,

a) run control specifications (user commands);

b) sewer system layout and physical characteristics of system components; and

c) inflow hydrographs at inlets, junctions, and manholes.

5.1.1. Run Control Specifications

The run control specifications, i.e., the commands describing the options to be used during the program execution, are declared on the first input card designated as START. This card is distinguished by the tag START b on its first six columns*. The options are separated by commas and/or blanks and can appear in any order. The options that can be specified are as follows.

NOEXEC - requests that the input data be verified, the operating environment for the numerical computations be prepared, but the numerical execution phase be bypassed. This option may be useful in checking inconsistencies and key-punching errors, or omissions, on the input deck of large sewer networks. To use this option the letters NOEXEC are punched on the START card.

DESIGN - requests the computation of the size of the sewers as described in Sec. 2.4. This option is specified by punching DESIGN on the START card.

---

*The underlined letter b is used to represent a blank space on the computer card.

NOGRAPH - requests that the computed discharge hydrographs not to be plotted with the standard line printer. To use this option, the letters NOGRAPH are punched on the START card.

Defaults for the above are EXEC, NODESIGN, and GRAPH; i.e., if they are not declared, the numerical computations will be carried on with the given sewer sizes and the computed discharge hydrographs will be plotted with the line printer.

COMPTST - requests the mini-compiler (subroutine COMPILE) which compiles and verifies the flow equation at the flow control structure (see outlet control card on p. 47) to produce extra diagnostic printout. This can be useful if the user suspects that the flow at the control structure is being evaluated incorrectly. The output enables the user to verify the exact sequence of operations followed in the evaluation of the flow equation at the control structure. It must be cautioned here that the use of COMPTST option may result in a significant amount of printout. The option is used simply by punching COMPTST on the START card.

The remaining operational commands on the START card are keyword codes and require the specification of a value.

TI = an integer value - specifies the time increment in sec, as given by the integer value, to be used in reading the inflow hydrograph ordinates. The computed discharge hydrographs and stage graphs are also printed at regular time increment TI. If this operational command does not appear in the START card, the default time increment is 30 sec.

MAXNODES = an integer value - specifies the number of nodes, i.e., the total number of inlets, junctions, manholes and the outlet. If the number of nodes in the system is less than 200, the user need not use this operational command.

DEBUG = 'options' - provides additional avenues primarily for program modification debugging purposes. This facility is generally not useful to an average user. An experienced programmer can use this facility to request for additional program output in order to detect and localize program errors. The options that can be specified within the apostrophes are stored internally as a string of characters, to be scanned later by a program section which subsequently print a prescribed diagnostic information. Options that can be specified typically consist of single key characters. The option DEBUG = 'I' which prints the initial flow conditions (which in fact is a default output) is given as an example in the program listing in Appendix A. Multiple options can be specified by multiple characters, e.g., DEBUG = 'DI'. Note that the apostrophes should be typed on the card.

GOPARM = 'options' - offers the ability to easily implement new options by the user without a significant reprogramming effort. This facility is implemented in a manner very similar to DEBUG. Multiple options can be specified within a pair of apostrophes by their names separated by a comma or a blank space. There are two options currently implemented via the GOPARM facility.

GOPARM = 'PLOT' - requests the computed discharge hydrographs and depth graphs at the sewer entrances to be written onto the disk data set represented by the //bCALCOMPbDD card (part of the JCL). This data can be read later for plotting by some suitable plotter.

GOPARM = 'INTVL = an integer value n' - specifies the frequency of print-out for the discharge, depth, and velocity along the sewers. This flow information is printed at every other n time steps (Δt), where the value of n is specified by the user. The default value of n is equal to 10.

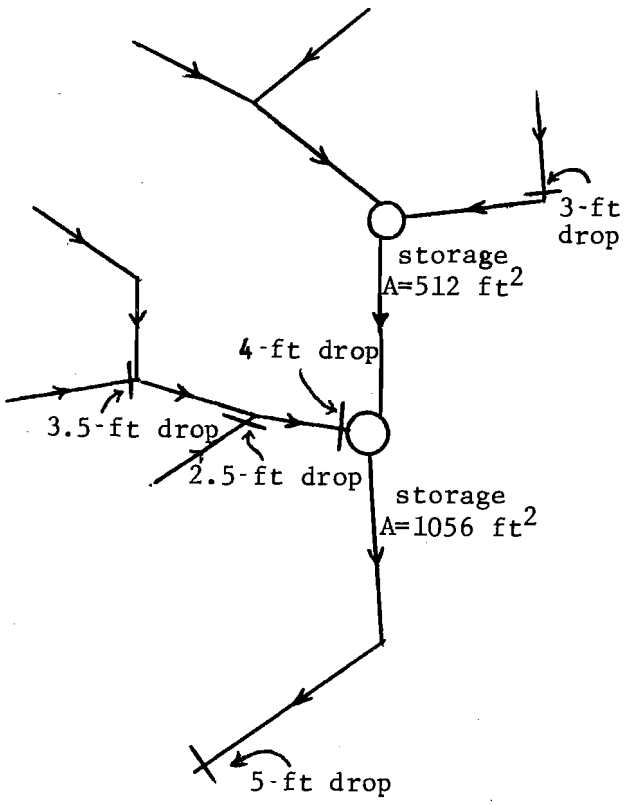
Some sample START cards are given below to illustrate various types of user commands.

```
START NOEXEC

START DESIGN TI = 100 GOPARM = 'PLOT, INTVL = 5'

START TI = 60 NOGRAPH GOPARM = 'PLOT' DEBUG = 'I' COMPTST

START TI = 120 GOPARM = 'INTVL = 100' DESIGN
```

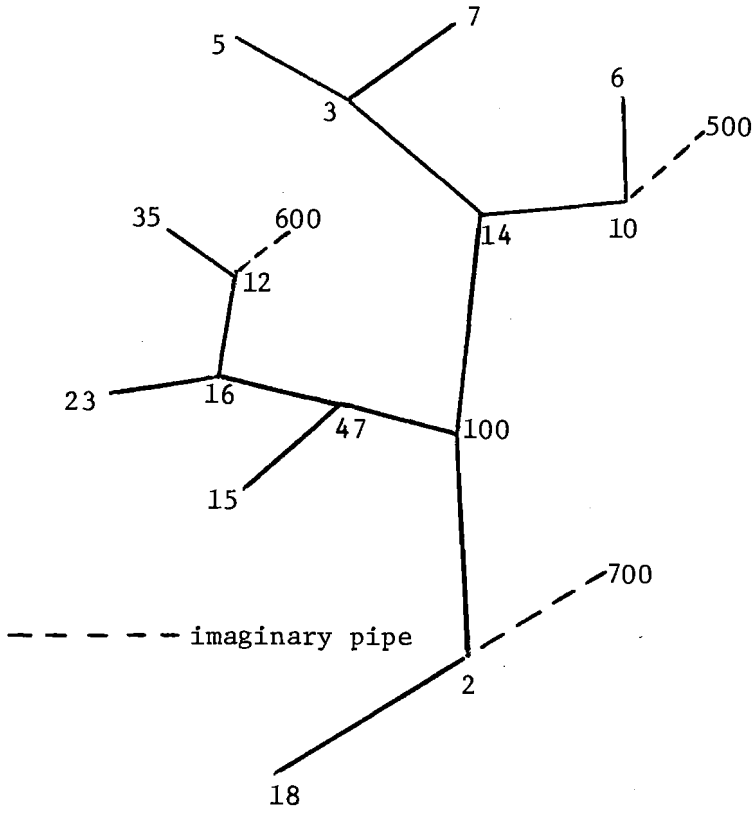
5.1.2. Sewer System Layout and Physical Characteristics of System Components

In the ISS Model a node-link representation is used to describe the layout (tree-structure) of the sewer system. Each inlet, junction, manhole, and the outlet of the sewer system is assigned an integer number (which need not follow any sequence) by the user. In order to make a two-way junction (Figs. 5a and 5c) "look like" a three-way junction (Figs. 5b and 5d), an imaginary pipe is assumed connecting to the former and hence an additional node number is specified at the upstream end of the imaginary pipe by the user. This first step in data preparation is schematically illustrated in Figs. 8a and 8b.

The node-link representation of the sewer system is then read into the computer through the following control cards.



a. Sewer layout



b. Node-link representation

PIPE GOES FROM NODE	LENGTH L, ft	SLOPE S_o,	ROUGHNESS k, ft	REMARKS
5	450	0.0030	0.0050	
7	600	0.0047	0.0050	
6	320	0.0080	0.0061	A 3-ft drop at the exit
500	0	0.0	0.0	Imaginary
3	800	0.0001	0.0040	
10	700	0.0002	0.0040	
35	510	0.0050	0.0060	
600	0	0.0	0.0	Imaginary
23	460	0.0090	0.0050	A 3.5-ft drop at the exit
12	360	0.0040	0.0050	
15	294	0.0093	0.0060	A 2.5-ft drop at the exit
16	380	0.0060	0.0040	A 2-ft drop at the exit
14	1000	0.0015	0.0060	Outflows from a storage element (A_j=512 ft²)
47	410	0.0045	0.0050	A 4-ft drop at the exit
100	1210	0.00015	0.0044	Outflows from a storage element (A_j= 1056 ft²)
700	0	0.0	0.0	Imaginary
2	1320	0.00098	0.0056	A 5-ft drop at the exit (outlet)

c. Physical characteristics of sewers (diameters to be computed)

FIG. 8. EXAMPLE SEWER SYSTEM

a) The NODE card - The user assigned node numbers are declared on the NODE cards which are distinguished by having the alphameric tag NODEb in the first five columns of each card. Following the tag the node numbers are punched, separated by one or more blanks or commas. The user may punch as many node numbers as will fit on the first 72 columns of a NODE card. Additional node numbers should be punched on subsequent NODE cards. Up to 200 nodes can be declared without specifying the MAXNODES value on the START card. The NODE card of the sewer system shown in Fig. 8 is given below as an example.

```
NODE 5 3 7 14 10 6 500
NODE 100 47 16 12 15 23 35 600
NODE 2 700 18
```

b) The IMAGINARY card - The user assigned node number at the upstream end of the imaginary sewers are declared on IMAGINARY cards which have the tag IMAGINARYb punched on the first ten columns of each card. The node numbers are entered in the same manner as on the NODE card. Thus, for the sewer system shown in Fig. 8,

```
IMAGINARY 500 600 700
```

c) The YJUNCT card - This card is used to specify which nodes represent the point-type (storageless) junctions in the sewer system. Following the tag YJUNCTb in the first seven columns, the junction node numbers are punched with the same format as in the NODE card. The YJUNCT card for the sewer system shown in Fig. 8 reads as follows.

```
YJUNCT 3 10 47 16 12
YJUNCT 2
```

d) The MANHOLE card - This card is used to specify the nodes which represent reservoir-type junctions, or storage elements, if any, of the sewer system. The tag MANHOLEb is punched on the first eight columns. Each

reservoir-type junction node is then declared on the card by the format AREA = n_1bn_2, where n_1 is the cross-sectional area of the reservoir in sq ft, and n_2 is the node number. A comma or one or more blanks should be used to separate different manholes. For instance, the MANHOLE card for the sewer system shown in Fig. 8 appears in the following form:

MANHOLE AREA = 512 14, AREA = 1056 100

e) The ROOT card - This card is used to identify the outlet of the sewer system. Following the tag ROOTb on the first five columns, the user-assigned node number to the system outlet is punched. The ROOT card for the sewer system shown in Fig. 8 is:

ROOT 18

f) The OUTLET CONTROL card - This card is used to describe the flow equation at the outlet control structure, if any. The tag OUTLETbCONTROLb is punched in the first 15 columns of the card. Following the tag, the initial value of flow depth (with reference to sewer bottom) at the system outlet is punched in any manner conforming to standard PL/1 (F) list-directed input rules. The remainder of the card contains the flow equation at the control, i.e., anyone of Eqs. 42, 43, 44 or 45 with the variable names DEPTH, DISCHARGE, VELOCITY, and TIME. The right hand side of the equation may include constants, operators of +, -, *, /, **, and the functions SIN, SIND, COS, COSD, and SQRT (SIN and SIND are alike, except the argument of SIN is in radians, whereas the argument of SIND is in degrees; the correspondence also applies to COS and COSD). In the case where the flow equation is too lengthy to fit on one OUTLET CONTROL card, it can be continued on up to two more continuation cards following immediately. These continuation cards are identified by a blank in column 1. Note that columns 73-80 of all cards are ignored. The OUTLET CONTROL card is processed and evaluated by a mini-compiler

(the subroutine COMPILE). This compiler possesses a peculiar sequence for evaluating functions, therefore one should use parentheses as grouping symbols everywhere in the flow equation where the sequence of operations matters. Some example OUTLET CONTROL cards are:

OUTLET CONTROL 5.25 DEPTH = 5.25
(Constant flow depth y = 5.25 ft)

OUTLET CONTROL 1. DEPTH = 1+(2*(SIN(6.28*TIME/42200)))
(y = 1 ft at t = 0, y = 1 + SIN(6.28t/42200) for t > 0)

OUTLET CONTROL 3. VELOCITY = 2*(DEPTH**1.5)
(y = 3 ft at t = 0, $V = 2y^{1.5}$)

OUTLET CONTROL 2.11 DISCHARGE = 10.5
(y = 2.11 ft at t = 0, pumpage at a rate Q = 10.5 cfs)

g) The PD card - Each pipe descriptor (PD) card defines the physical characteristics of a sewer in the system. The tag PDb in the first three columns of the card is followed by eight numerical values specifying in the following order:

(i) node number at the sewer entrance;

(ii) node number at the sewer exit;

(iii) sewer length, L, (ft);

(iv) sewer slope, S_o, (%);

(v) sewer diameter, D, (ft)*;

(vi) effective roughness, k, of sewer, (ft);

(vii) height of drop at sewer exit, Z, (ft)**; and

(viii) computational space increment, Δx, (ft)***.

---

*If the DESIGN option has been specified (on the START card), the diameter is declared to be zero on the PD card.

**If there exists no drop at the sewer exit, the height of the drop is zero on the PD card.

***The value of Δx should not be greater then L/2, and it should be selected so as to ensure that $L/\Delta x$ is an integer.

The first two numbers are integers while the remaining numbers can be punched in any manner conforming to standard PL/1 (F) list-directed input rules; e.g., any of the following types are equally acceptable: 1, 1., 1.00, .2, 0.02, 1E-5, 3.52, 4E10. Blanks and/or commas are used to delimit the numerical values. The PD cards for the sewers of the network shown in Fig. 8 are listed below as an example.

```
PD   5   3   450   .003   0   .005   0   90
PD   7   3   600   .0047   0   .005   0   60
PD   3   14   800   .0001   0   .004   0   100
PD   6   10   320   .008   0   .0062   3   80
PD   10   14   700   .0002   0   .004   0   100
PD   500   10   0   0   0   0   0   0
PD   14   100   1000   .0015   0   .006   0   125
PD   47   100   410   .0045   0   .005   4.1   82
PD   15   47   294   .0093   0   .006   2.5   98
PD   16   47   380   .006   0   .004   2   76
PD   23   16   460   .009   0   .005   3.5   92
PD   12   16   360   .004   0   .005   0   90
PD   35   12   520   .005   0   .006   0   104
PD   600   12   0   0   0   0   0   0
PD   100   2   1210   .00015   0   .0045   0   121
PD   700   2   0   0   0   0   0   0
PD   2   18   1320   .00098   0   .0057   5   132
```

5.1.3. Inflow Hydrographs

The inflow hydrographs at those nodes representing the inlets of the sewer system are specified as follows:

a) Node number, time (τ, in sec) when the inflow hydrograph starts*, duration (T, in sec) of the hydrograph, and the magnitude of the initial baseflow (Q_b in cfs) are declared in that order on the FBD (FLOWBLOCK descriptor) card. Following the tag FBDb in the first four columns, the first three integer values for node number, τ, and T are punched while the

*The time, τ, at which the inflow hydrograph starts at a particular node is defined relative to the time at which the earliest inflow hydrograph of the sewer system starts. In other words, the earliest occurring inflow hydrograph is assumed to start at time $\tau = 0$. The starting time of the remaining inflow hydrographs are then specified accordingly with respect to this time scale.

last one for Q_b may be punched in any acceptable form for PL/1 list-directed input (see examples in the description of the PD card in Fig. 10). The numerical values are delimited by one or more blanks.

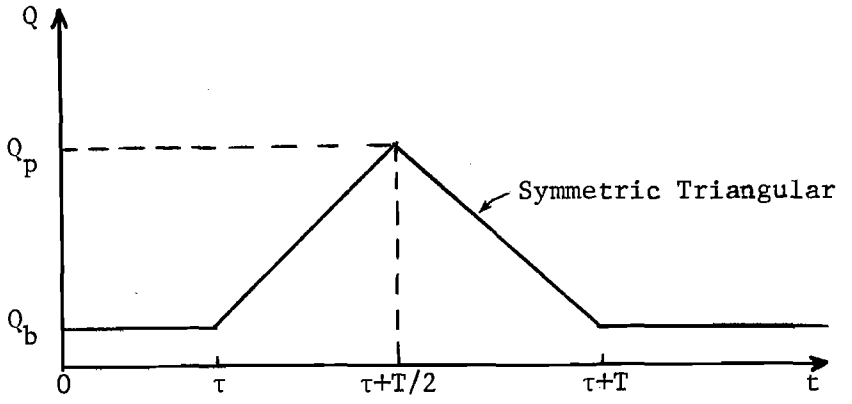
b) Following the FBD card are one or more cards on which pairs of the time - discharge values of the inflow hydrograph are punched within columns 1 to 72. Obviously, the first pair is the starting time (τ) - baseflow (Q_b), and the last pair is the time ($\tau + T$) - baseflow (Q_b). Although the ordinates of the inflow hydrograph can be read in at irregular time intervals within the period $\tau \leq t \leq \tau + T$, they are stored at regular time interval TI which is specified on the START card. Thus, it is desirable to select TI such that τ/TI and T/TI are integers for all the inflow hydrographs. The ordinates of the inflow hydrograph outside the period $\tau \leq t \leq \tau + T$ which are not read in are automatically set to the initial baseflow value, Q_b, by the program.

c) For each inflow hydrograph, the data cards described above are succeeded by a FEND card to indicate the end of the hydrograph. The FEND card has the tag FEND punched on its first four columns.

d) The direct storm water inflow into or pumpage out from the junctions is specified in exactly the same manner as the inflow hydrographs at the inlets. However, the tags FBDb and FEND are now replaced by the tags JFBDb (Junction FLOWBLOCK descriptor) and JEND, respectively.

The data cards describing the hypothetical design inflow hydrographs shown in Fig. 9 for the example sewer system (Fig. 8) are listed in the following as an example.

```
FBD  5  0  960  1
0 1, 120 3, 240 5, 360 7, 480 9, 600 7, 720 5, 840 3, 960 1
FEND
FBD  7  120  720  0.5
120 0.5, 240 2, 360 3.5, 480 5, 600 3.5, 720 2.0, 840 0.5
FEND
FBD  6  240  1200  1
```



INPUT NODE (see Fig. 7)	TIME LAG τ, sec.	DURATION T, sec.	BASEFLOW Q_b, cfs	PEAKFLOW Q_p, cfs
5	0	960	1.0	9
7	120	720	0.5	5
6	240	1200	1.0	11
15	240	1200	1.0	11
23	360	1200	1.0	11
35	480	1200	1.0	11
14	120	960	1.0	9
12	0	720	0.5	5
2	600	1200	1.0	11

FIG. 9. HYPOTHETICAL DESIGN INFLOW HYDROGRAPHS FOR EXAMPLE SEWER SYSTEM

```
240 1, 360 3, 480 5, 600 7, 720 9, 840 11, 960 9, 1080 7, 1200 5,
 1320 3, 1440 1
FEND
FBD  15  240  1200  1
240 1, 360 3, 480 5, 600 7, 720 9, 840 11, 960 9, 1080 7, 1200 5, 1320
 3, 1440 1
FEND
FBD  23  360  1200  1
360 1, 480 3, 600 5, 720 7, 840 9, 960 11, 1080 9, 1200 7, 1320 5,
 1440 3, 1560 1
FEND
FBD  35  480  1200  1
480 1, 600 3, 720 5, 840 7, 960 9, 1080 11, 1200 9, 1320 7, 1440 5,
 1560 3, 1680 1
FEND
JFBD  14  120  960  1
120 1, 240 3, 360 5, 480 7, 600 9, 720 7, 840 5, 960 3, 1080 1
JEND
JFBD  12  0  720  0.5
0. 0.5, 120 2, 240 3.5, 360 5, 480 3.5, 600 2, 720 0.5
JEND
JFBD  2  600  1200  1
600 1, 720 3, 840 5, 960 7, 1080 9, 1200 11, 1320 9, 1440 7, 1560 5,
 1680 3, 1800 1
JEND
```

The last card of the input data deck is the CEND card. This card has the tag CEND punched on its first four columns. It is used to separate the input decks of the different sewer systems in case several operation or design studies are to be made in a single computer run.

5.2. Program Output

The output from the ISS Model falls basically into two classes. They are the output generated during the input deck read-in/verification phase and the output generated during the numerical simulation phase.

5.2.1 Output From Read-in/Verification Phase

The first printed output from the program is a listing of the input deck. Each input card is listed exactly as it was read in, followed by an acknowledgment line printed by the program giving the information which may be examined to verify the correctness of the program's scanning of the input

card. The printed acknowledgment also helps to clarify the extent to which the input cards have been processed when an error is detected.

During the read-in portion of the read-in/verification phase, the input cards are examined by the program for information conflicts. For example, it would be a conflict-type error if a particular node number appeared on both a ROOT card and a YJUNCT card of the sewer network. Errors of this type are generally "fatal"; i.e., the program prints an appropriate (self explanatory) error diagnostic message, terminates the execution for that particular sewer network, and proceeds to read in the input deck of the next sewer network, if any.

During the verification portion of the read-in/verification phase, the input deck is examined by the program for information omissions. For example, missing of one of the pipe descriptor (PD) cards would be an omission-type error. Errors of this type are always "fatal", but often the processing may continue to find other omission-type errors, and prints out appropriate warning and error messages, before terminating the execution for that particular sewer network.

Upon successful completion of the read-in/verification phase, an appropriate built-in message is printed out and the physical characteristics of all the sewers in the system are listed in a tabular form as illustrated in Fig. 10 for the example sewer system show in Fig. 8.

5.2.2. Output From Numerical Simulation Phase (Flow Information)

As described in Sec. 2.2 , for numerical solution the network is divided into a number of overlapping Y-segments. Following an appropriate sequence these Y-segments are solved one at a time. Therefore, the output from the numerical simulation phase of the program is printed out for each Y-segment, one segment at a time, following the order they are computed.

The output for each segment consists of:

a) discharge hydrograph and depth graph at the entrance to the sewers;

b) variations of the discharge, depth, and velocity at the computational grid points along the sewers at prescribed time intervals (see the option GOPARM = 'INTVL = an integer value' in Subsection 5.1.1).

c) plots of discharge hydrograph at sewer entrances by the line printer, unless the NOGRAPH option is specified on the START card; and

d) design information, if the DESIGN option has been specified on the START card. As described in Sec. 2.4, the program searches in a systematic way for the most suitable commercially available pipe sizes. The trial pipe sizes during this search is printed as the design information. Upon completion of the DESIGN run, the computed diameters of all the sewers in the system is further listed in a tabular form.

Sample outputs from the numerical simulation phase of the computer program for the example sewer system shown in Fig. 8 are given in Figs. 11 and 12. The discharge hydrographs and depth graphs shown in Fig. 12 are plotted by using the CALCOMP plotting program listed in Appendix C. Most computer installations have made some modifications to the basic CALCOMP subroutines. Therefore the plotting program given in Appendix C should be modified accordingly, or a new plotting program should be written if a graphic display of flow conditions is desired.

```
DATE 09/22/73          UNIVERSITY OF ILLINOIS CIVIL ENGINEERING--STORM SEWER SIMULATION PROGRAM

                                CONTROL CARD LISTING FOR STORM SEWER NETWORK # 1

                    START TI=120 GOPARM=' INTVL=50 ' NOGRAPH DESIGN

                    NODE 5 3 7 14 10 6 500
*NODES DECLARED: 5 3 7 14 10 6 500

                    NODE 100 47 16 12 15 23 35 600
*NODES DECLARED: 100 47 16 12 15 23 35 600

                    YJUNCT 3 10 47 16 12
*Y-JUNCTIONS: 3 10 47 16 12

                    NODE 2 700 18
*NODES DECLARED: 2 700 18

                    IMAGINARY 500 600 700
*IMAGINARY NODES: 500 600 700

                    PD 5 3 450 .003    0 .005 0 90
*PIPE GOES FROM NODE #5 TO NODE #3

                    PD 7 3 600 .0047    0 .005 0 60
*PIPE GOES FROM NODE #7 TO NODE #3

                    PD 3 14 800 .0001    0 .004 0 100
*PIPE GOES FROM NODE #3 TO NODE #14

                    PD 6 10 320 .008    0 .0062 3 80
*PIPE GOES FROM NODE #6 TO NODE #10

                    PD 10 14 700 .0002    0 .004 0 100
*PIPE GOES FROM NODE #10 TO NODE #14

                    PD 500 10 0 0  0 0 0
*PIPE GOES FROM NODE #500 TO NODE #10

                    PD 14 100 1000 .0015    0 .006 0 125
*PIPE GOES FROM NODE #14 TO NODE #100

                    PD 47 100 410 .0045    0 .005 4.1 82
*PIPE GOES FROM NODE #47 TO NODE #100

                    PD 15 47 294 .0093    0 .006 2.5 98
*PIPE GOES FROM NODE #15 TO NODE #47

                    PD 16 47 380 .006    0 .004 2 76
*PIPE GOES FROM NODE #16 TO NODE #47

                    PD 23 16 460 .009    0 .005 3.5 92
*PIPE GOES FROM NODE #23 TO NODE #16

                    PD 12 16 360 .004    0 .005 0 90
*PIPE GOES FROM NODE #12 TO NODE #16

                    PD 35 12 520 .005    0 .006 0 104
*PIPE GOES FROM NODE #35 TO NODE #12

                    PD 600 12 0 0  0 0 0
*PIPE GOES FROM NODE #600 TO NODE #12

                    PD 100 2 1210 .00015    0 .0045 0 121
*PIPE GOES FROM NODE #100 TO NODE #2

                    PD 2 18 1320 .00098    0 .0057 5 132
*PIPE GOES FROM NODE #2 TO NODE #18

                    PD 700 2 0 0 0 0 0
*PIPE GOES FROM NODE #700 TO NODE #2

                    MANHOLE AREA=512 14, AREA=1056 100
*MANHOLES: 14 100
```

FIG. 10. PROGRAM OUTPUT FROM READ IN/VERIFICATION PHASE

```
                    FBD 5 0 960 1
*FLOWBLOCK DESCRIPTION FOR NODE # 5.  (TIME LAG, DURATION, BASE FLOW.)
                    0 1,120 3,240 5,360 7,480 9,600 7,720 5,840 3,960 1
                    FEND

                    FBD 7 120 720 0.5
*FLOWBLOCK DESCRIPTION FOR NODE # 7.  (TIME LAG, DURATION, BASE FLOW.)
                    120 0.5,240 2,360 3.5,480 5,600 3.5,720 2.0,840 0.5
                    FEND

                    FBD 6 240 1200 1
*FLOWBLOCK DESCRIPTION FOR NODE # 6.  (TIME LAG, DURATION, BASE FLOW.)
                    240 1,360 3,480 5,600 7,720 9,840 11,960 9,1080 7,1200 5,1320 3,1440 1
                    FEND

                    FBD 15 240 1200 1
*FLOWBLOCK DESCRIPTION FOR NODE # 15.  (TIME LAG, DURATION, BASE FLOW.)
                    240 1,360 3,480 5,600 7,720 9,840 11,960 9,1080 7,1200 5,1320 3,1440 1
                    FEND

                    FBD 23 360 1200 1
*FLOWBLOCK DESCRIPTION FOR NODE # 23.  (TIME LAG, DURATION, BASE FLOW.)
                    360 1,480 3,600 5,720 7,840 9,960 11,1080 9,1200 7,1320 5,1440 3,1560 1
                    FEND

                    FBD 35 480 1200 1
*FLOWBLOCK DESCRIPTION FOR NODE # 35.  (TIME LAG, DURATION, BASE FLOW.)
                    480 1,600 3,720 5,840 7,960 9,1080 11,1200 9,1320 7,1440 5,1560 3,
                     1680 1
                    FEND

                    JFBD 14 120 960 1
*FLOWBLOCK DESCRIPTION FOR NODE # 14.  (TIME LAG, DURATION, BASE FLOW.)
                    120 1,240 3,360 5,480 7,600 9,720 7,840 5,960 3,1080 1
                    JEND

                    JFBD 12 0 720 0.5
*FLOWBLOCK DESCRIPTION FOR NODE # 12.  (TIME LAG, DURATION, BASE FLOW.)
                    0 0.5,120 2,240 3.5,360 5,480 3.5,600 2,720 0.5

                    JEND

                    JFBD 2 600 1200 1
*FLOWBLOCK DESCRIPTION FOR NODE # 2.  (TIME LAG, DURATION, BASE FLOW.)
                    600 1,720 3,840 5,960 7,1080 9,1200 11,1320 9,1440 7,1560 5,1680 3,1800
                    1
                    JEND

                    YJUNCT 2
*Y-JUNCTIONS: 2

                    ROOT 18

                    CEND
```

```
*NODE VALIDITY CHECKING BEGINS.
*WARNING* NO FLOW WAS SPECIFIED INTO IMAGINARY NODE, #   500.
*WARNING* NO FLOW WAS SPECIFIED INTO IMAGINARY NODE, #   600.
*WARNING* NO FLOW WAS SPECIFIED INTO IMAGINARY NODE, #   700.
*NODE VALIDITY CHECKING HAS COMPLETED SUCCESSFULLY.  TREE VALIDITY CHECKING BEGINS.
*DELETING NODES:     5     7     6    500    15    23    35   600   700    18     3    10    12     2    14
    100    47
*TREE VALIDITY CHECKING HAS COMPLETED SUCCESSFULLY.  READY FOR FLOW EVALUATION.
```

FIG. 10. (Continued)

PIPES SUMMARY TABLE

PIPE GOES FROM_NODE	TO_NODE	LENGTH IN_FEET	SLOPE	DIAMETER IN_FEET	ROUGHNESS	DROP IN_FEET	DELTA_X IN_FEET
5	3	450	.002999	0.00	.0049	0.0	90
7	3	600	.004699	0.00	.0049	0.0	60
6	10	320	.007999	0.00	.0061	3.0	80
500	10	1	.000000	1.00	.0000	0.0	1
3	14	800	.000099	0.00	.0039	0.0	100
10	14	700	.000199	0.00	.0039	0.0	100
35	12	520	.004999	0.00	.0059	0.0	104
600	12	1	.000000	1.00	.0000	0.0	1
23	16	460	.008999	0.00	.0049	3.5	92
12	16	360	.003999	0.00	.0049	0.0	90
15	47	294	.009299	0.00	.0059	2.5	98
16	47	380	.005999	0.00	.0039	2.0	76
14	100	1,000	.001499	0.00	.0059	0.0	125
47	100	410	.004499	0.00	.0049	4.0	82
100	2	1,210	.000149	0.00	.0044	0.0	121
700	2	1	.000000	1.00	.0000	0.0	1
2	18	1,320	.000979	0.00	.0056	5.0	132

FIG. 10. (Concluded)

INPUT HYDROGRAPH SPECIFIED FOR NODE # 5

INCREMENT #	ABS. TIME, SEC	DISCHARGE, CFS
0	0.0	1.0
1	120.0	3.0
2	240.0	5.0
3	360.0	7.0
4	480.0	9.0
5	600.0	7.0
6	720.0	5.0
7	840.0	3.0
8	960.0	1.0

INPUT HYDROGRAPH SPECIFIED FOR NODE # 7

INCREMENT #	ABS. TIME, SEC	DISCHARGE, CFS
1	120.0	0.5
2	240.0	2.0
3	360.0	3.5
4	480.0	5.0
5	600.0	3.5
6	720.0	2.0
7	840.0	0.5

FLOW CONDITIONS AT TIME = 1,736.7 SECONDS

INTERIOR STATION NUMBER	FROM NODE 5 TO NODE 3			FROM NODE 7 TO NODE 3			FROM NODE 3 TO NODE 14		
	VELOCITY	DEPTH	DISCHARGE	VELOCITY	DEPTH	DISCHARGE	VELOCITY	DEPTH	DISCHARGE
0	2.43	0.40	1.0	2.33	0.25	0.5	0.35	1.06	0.9
1	2.42	0.40	1.0	2.33	0.25	0.5	0.37	1.07	1.0
2	2.32	0.42	1.0	2.33	0.25	0.5	0.40	1.08	1.0
3	1.58	0.56	1.0	2.33	0.25	0.5	0.42	1.09	1.1
4	0.91	0.80	1.0	2.33	0.25	0.5	0.45	1.10	1.2
5	0.63	1.06	0.9	2.33	0.25	0.5	0.47	1.10	1.3
6				2.33	0.25	0.5	0.50	1.11	1.4
7				2.32	0.25	0.5	0.53	1.12	1.5
8				-0.09	0.50	-0.1	0.55	1.13	1.5
9				-0.03	0.78	-0.0			
10				-0.01	1.06	-0.0			

INPUT HYDROGRAPH CALCULATED FOR NODE # 3

ABS. TIME, SEC	DEPTH, FEET	DISCHARGE, CFS
120.0	0.87	1.3
240.0	1.01	3.4
360.0	1.24	6.8
480.0	1.49	10.0
600.0	1.64	10.7
720.0	1.62	8.4
840.0	1.53	5.6
960.0	1.39	3.0
1,080.0	1.28	1.6
1,200.0	1.23	1.5
1,320.0	1.18	1.4
1,440.0	1.15	1.3
1,560.0	1.11	1.1
1,680.0	1.08	0.9

FIG. 11. SAMPLE OUTPUTS FROM NUMERICAL SIMULATION PHASE

PIPES SUMMARY TABLE

PIPE GOES FROM_NODE	TO_NODE	LENGTH IN_FEET	SLOPE	DIAMETER IN_FEET	ROUGHNESS	DROP IN_FEET	DELTA_X IN_FEET
5	3	450	.002999	1.75	.0049	0.0	90
7	3	600	.004699	1.75	.0049	0.0	60
6	10	320	.007999	1.75	.0061	3.0	80
500	10	1	.000000	1.00	.0000	0.0	1
3	14	800	.000099	3.50	.0039	0.0	100
10	14	700	.000199	3.00	.0039	0.0	100
35	12	520	.004999	1.75	.0059	0.0	104
600	12	1	.000000	1.00	.0000	0.0	1
23	16	460	.008999	1.50	.0049	3.5	92
12	16	360	.003999	1.75	.0049	0.0	90
15	47	294	.009299	1.50	.0059	2.5	98
16	47	380	.005999	2.25	.0039	2.0	76
14	100	1,000	.001499	3.25	.0059	0.0	125
47	100	410	.004499	2.75	.0049	4.0	82
100	2	1,210	.000149	5.00	.0044	0.0	121
700	2	1	.000000	1.00	.0000	0.0	1
2	18	1,320	.000979	3.50	.0056	5.0	132

FIG. 11. (Continued)

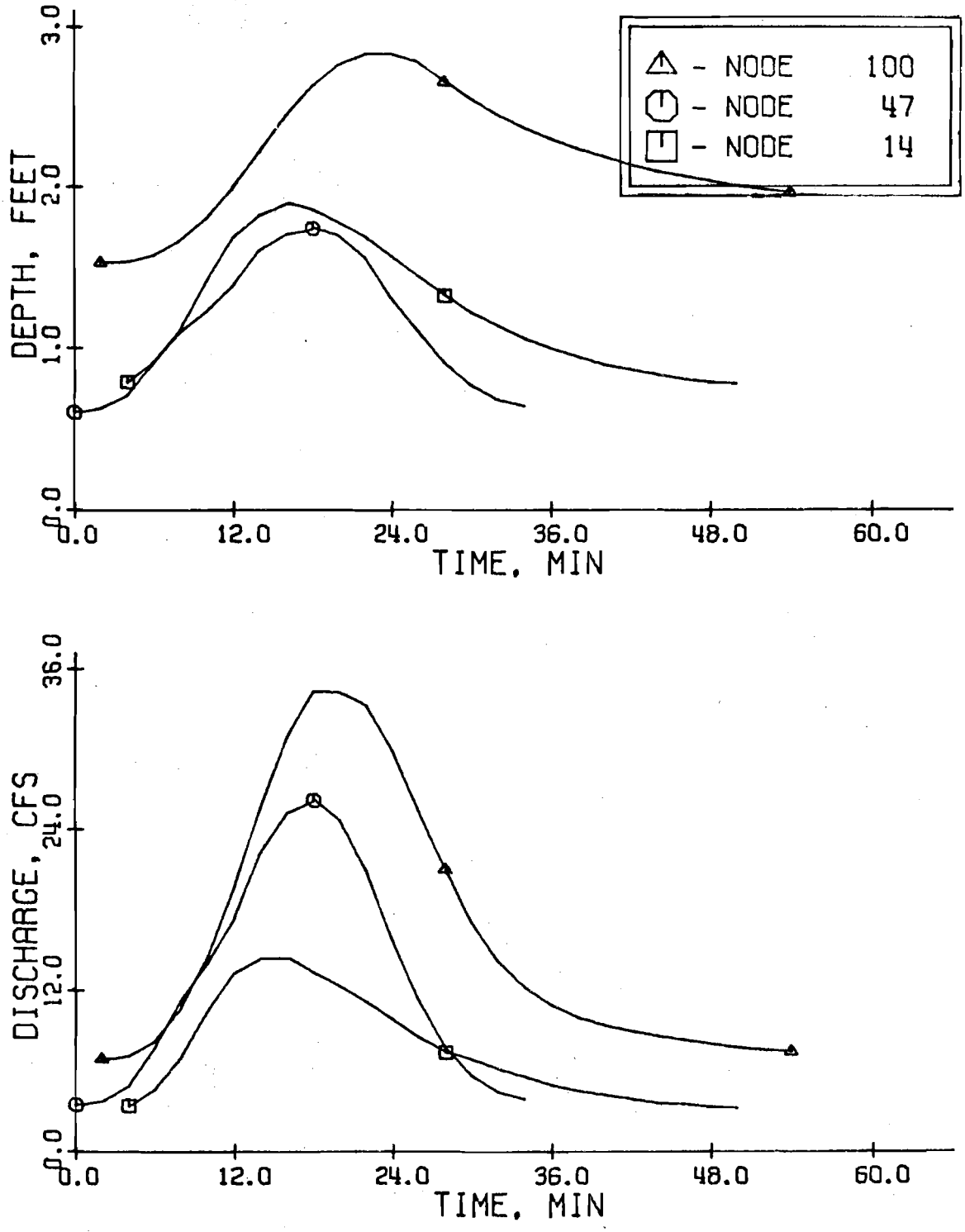


FIG. 12. SAMPLE CALCOMP PLOTS AT SEWER ENTRANCES

6. STRUCTURE OF COMPUTER PROGRAM

6.1. Program Structure

The computer program of the ISS Model consists of a main controlling section, NETWORK, an input data and control card scanner and data structure constructer, SETUP, a recurrsive Y-segment sequencing director, SRCHTREE, and a numerical simulation routine, EVALU8R. The linkage of these major routines are illustrated in Fig. 13. Detailed block diagram of the program is given in Appendix A.

The main controlling section NETWORK is a calling program which calls the routine SETUP repeatedly until all the input cards have been read in. The SETUP routine, in addition to reading in all of the input cards, determines the types of the cards, processes their operational codes, and creates the internal data structures to represent the sewer network system. During these procedures it performs exhaustive error checking to ensure that there is no conflict or omission type errors in the input data. If an error is detected it prints an appropriate diagnostic warning or error message by calling on one of its several subroutines.

Upon the completion of the read-in and verification procedures, the control is returned from SETUP to NETWORK. The NETWORK section then calls for SRCHTREE. The recurrsive routine SRCHTREE searches through the sewer network until it finds a Y-segment for which the upstream inflow hydrographs, referred to as FLOWBLOCKS in the program, are known. Having found such a Y-segment SRCHTREE passes the control to EVALU8R.

The EVALU8R routine first performs various initialization procedures. These include expansion of each of the two known FLOWBLOCKS so that they cover identical time period, creation of another FLOWBLOCK of the same size to hold the inflow hydrograph of downstream sewer of the Y-segment, and dynamic creation of work arrays, etc. EVALU8R then performs

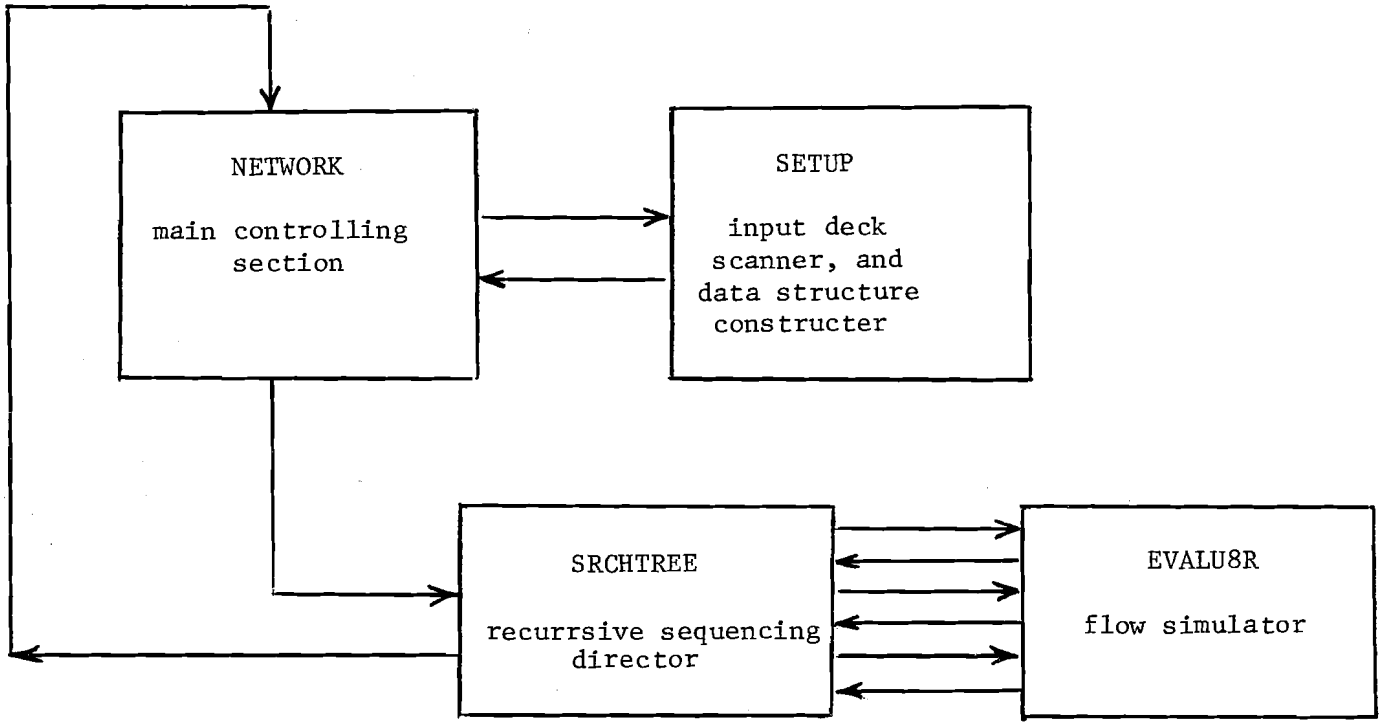


FIG. 13. BLOCK DIAGRAM OF COMPUTER PROGRAM

the numerical simulation by calling its various subroutines. After completion of the numerical computations of the current Y-segment, EVALU8R returns the two upstream FLOWBLOCKS and all work arrays, which are no longer needed, to the memory pool so that these areas of memory can be re-used later. Then the control is passed to SRCHTREE.

Upon return to SRCHTREE, the junction node of the Y-segment just computed now appears as an input node. With each completion of the EVALU8R cycle, therefore, two upstream sewers of the Y-segment are logically removed from the sewer network. By using this very efficient recurrsive algorithm the sewer network is "pruned" successively until it is reduced to the last Y-segment at the outlet. After the last Y-segment is evaluated, the routine EVALU8R returns the control to the main program, NETWORK, via the routine SRCHTREE. NETWORK then returns all the remaining memories used by the completed sewer network back to the memory pool. Then it reads in the input data and control cards for the next sewer network, if any, to re-start the cycle of the procedures. The program operation is terminated when there is no more sewer network input deck to be processed.

6.2. Internal Data Structures

Because of the tree-type network of the sewer system considered, it is especially important to select a proper list-processing technique for data handling. In this section the data structures of the information units NODE and FLOWBLOCK are described. The former is for the data concerning the network components. The latter is for hydrographs.

6.2.1. Information Unit NODE

In order to equally represent any arbitrary sewer network, a uniform information unit is desirable. The information unit, NODE, is used for each inlet, junction (YJUNC in the program), manhole (MANHOLE), and

outlet (ROOT) in the network. Each NODE consists of a "packet" of information containing the following:

a) node number - the user-assigned number to identify the node;

b) flags indicating the node type, e.g., YJUNCT, IMAGINARY MANHOLE, ROOT;

c) pointer variables pointing to the immediately adjacent upstream and downstream nodes;

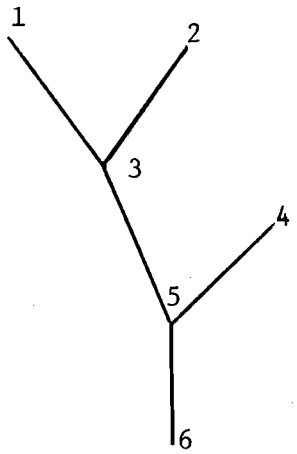
d) pointers to find FLOWBLOCK associated with the node; and

e) information regarding the sewer which emanates from the node, i.e., the length, diameter, slope, roughness, height of drop at sewer exit, and the computational parameter Δx to be used in the numerical computations.

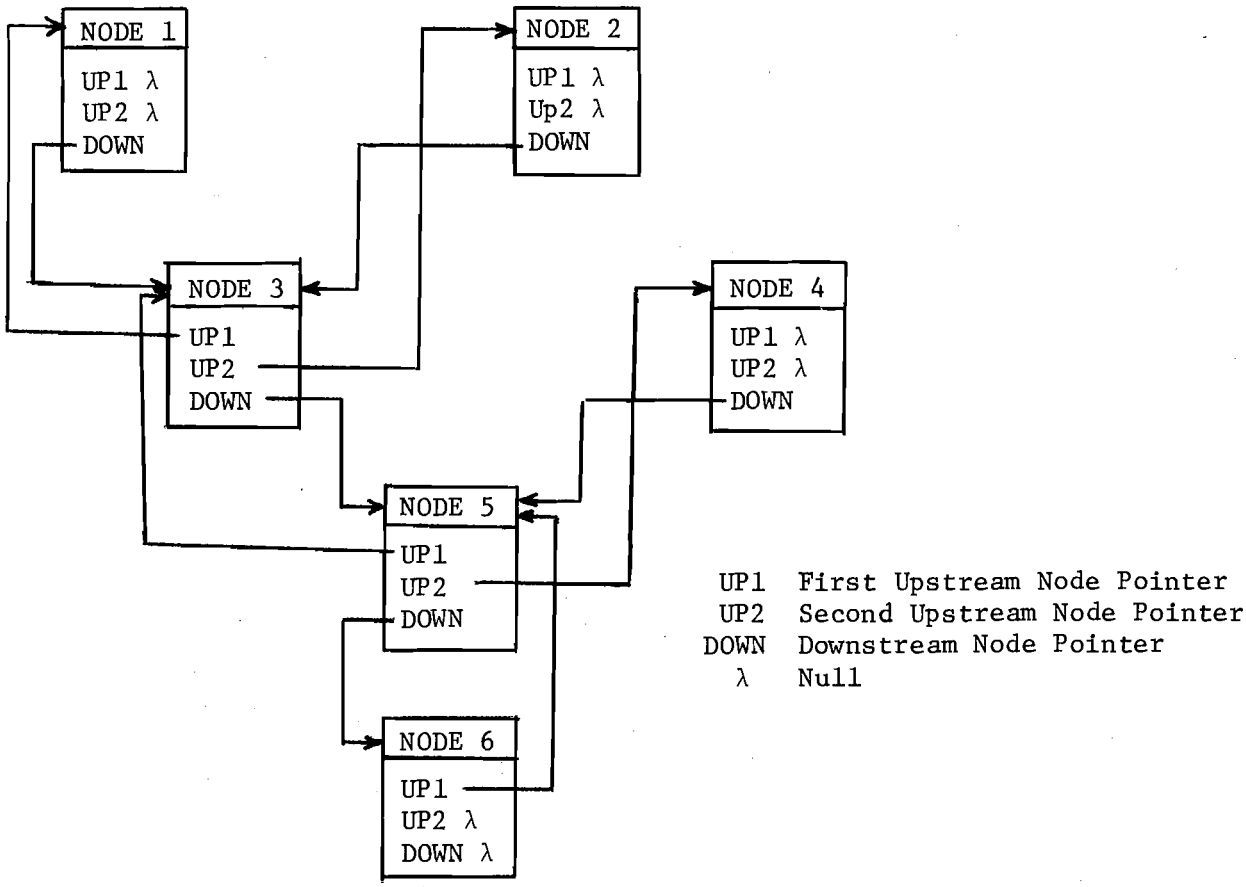
The linkage of information units to form a network is illustrated in Fig. 14b for the example small network shown in Fig. 14a. As can be seen from Fig. 14a, the information units NODE 1, 2 and 4 representing the inlets of the network do not contain pointers pointing to any upstream NODE, and NODE 6 representing the outlet (ROOT NODE) does not contain pointers pointing to any downstream NODE. It is important to realize in this technique of network connectivity that it is possible for the segment consisting of the nodes 1, 2 and 3 to "look like" the segment consisting of the nodes 3, 4 and 5. This provides an exceedingly simple recurrsive technique of the method of overlapping segments for dealing with arbitrary tree-type networks as will be described in Subsection 6.4.3.

6.2.2. Information Unit FLOWBLOCK

As mentioned previously, hydrograph data are stored in FLOWBLOCKS. A distinct problem in a multi-element routing like the ISS Model is how to keep track of the information concerning the hydrographs such as FLOWBLOCKS in the present program. At any given point of execution there may be literally any number of these FLOWBLOCKS, and the time period (in real time) covered by each of them may be quite different. To overcome this difficulty,



a. Schematic of sewer layout



b. NODE linkage

FIG. 14. NODE LINKAGE AND NETWORK CONNECTIVITY

FLOWBLOCKS are chosen to be dynamically sized arrays. The number of elements in each FLOWBLOCK may be different, and they are determined when the FLOWBLOCK is created. The lower and upper bounds of each FLOWBLOCK can vary independently so as to allow the FLOWBLOCK to expand, compact, and "slide" along the time axis. Thus, FLOWBLOCKS contain all the information of interest without wasting memory spaces (Fig. 15). The program can determine the real time covered by any FLOWBLOCK simply by checking the bounds representing the range of time, τ/TI and (τ+T)/TI. The discharge values falling outside of the time range covered by a FLOWBLOCK is automatically set equal to the baseflow rate.

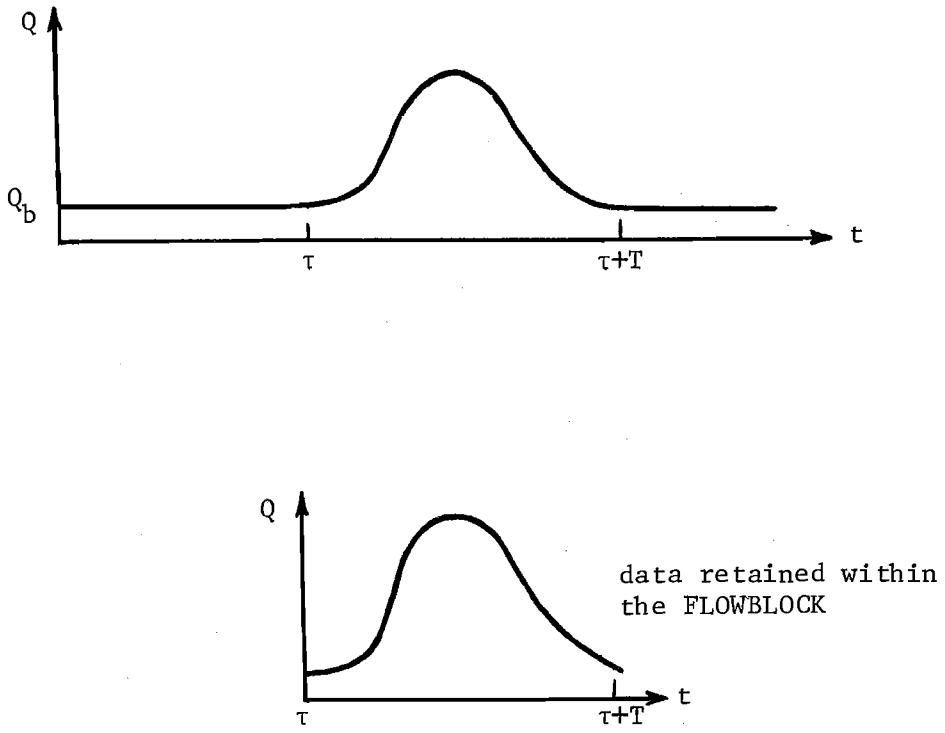


FIG. 15. FLOWBLOCK

6.3. PL/1 Controlled Variables Referencing and Stacking

In PL/1 language, controlled variables can be allocated and freed under program control. Essentially anything that can be specified for automatic variables can be used in conjunction with controlled variables. Particularly important is that both upper and lower bounds of a controlled array can be specified as variable functions at the time of allocation. This is in contrast with the case of based variables, the dimensionality of which is more restricted; e.g., with a PL/1 (F) based array, only the upper bound may vary through the use of inefficient REFER option. The extra flexibility of controlled variables stems from the fact that each allocation of a controlled array is attached by its own copy of the array dope vector describing that particular allocation (or generation) of the controlled array.

In addition, each allocation of controlled array also has at its beginning a three- or four-word controlled variable block; this includes a pointer to the immediately preceding allocation of the controlled variable, if any. All of the allocations of a given controlled variable together forms a normal linked list. The most recent allocation of the variable, normally at the "top" of the linked list, is pointed to by a PSEUDOREGISTER contained in the PL/1 PSEUDOREGISTER VECTOR, or PRV.

The major storage management trick used in the program is that with a short assembly language routine, one has a pointer which locates the pseudoregister for any desired controlled variable. This trick gives the programmer almost direct access to any allocations of the related controlled variable by simply "stuffing" the proper pointer into the pseudoregister. The pointer to be "stuffed in" has been saved earlier immediately after the original allocation of the controlled array which is now to be accessed.

It is important to note that in order to maintain the integrity of the controlled variable stack, no allocations for a controlled variable can be made unless the pseudoregister points at the most recent allocation. Likewise, no allocations can be freed unless the same holds true. It is possible, however, to free any desired generation of a controlled variable by rearranging the controlled variable stack (by manipulating the chain pointers) so that the allocations to be freed "look like" the most recently allocated generation. In the program this stack rearrangement is performed by the PL/1 subroutine "PLUCK".

6.4. Programming Techniques

6.4.1. Management of Work Arrays

In order to carry out the numerical operation a series of temporary working storage arrays is needed. These arrays store the values of discharge, velocity and depth at each space increment, Δx, along the sewers of a Y-segment. Two sets of three arrays are necessary for each sewer, one set for the current values of Q, V, and y at the time t, and the other for the new values of Q, V and y at the time $t + \Delta t$. The size of the six arrays associated with each sewer is the same, but the size may vary for different sewers. The lower bound (beginning subscript) of the six work arrays for a sewer is a constant. The upper bound is computed as a function of the sewer length, L, and space increment, Δx, which have been specified on the PD card.

The work arrays are named Qn, Dn, Vn, QDTn, DDTn and VDTn, where Q, D and V denote discharge, depth, and velocity, respectively; DT indicates the "new values" at $t = t + \Delta t$, and the integer n = 1, 2, or 3 identifies the sewer in the segment (Fig. 16). These work arrays are implemented as simple PL/1 controlled arrays. They are allocated via standard PL/1 ALLOCATE statements shortly after the EVALU8R subroutine is invoked, and they are

freed by standard PL/1 FREE statements just before the EVALU8R subroutine returns to the invoking procedure, SRCHTREE.

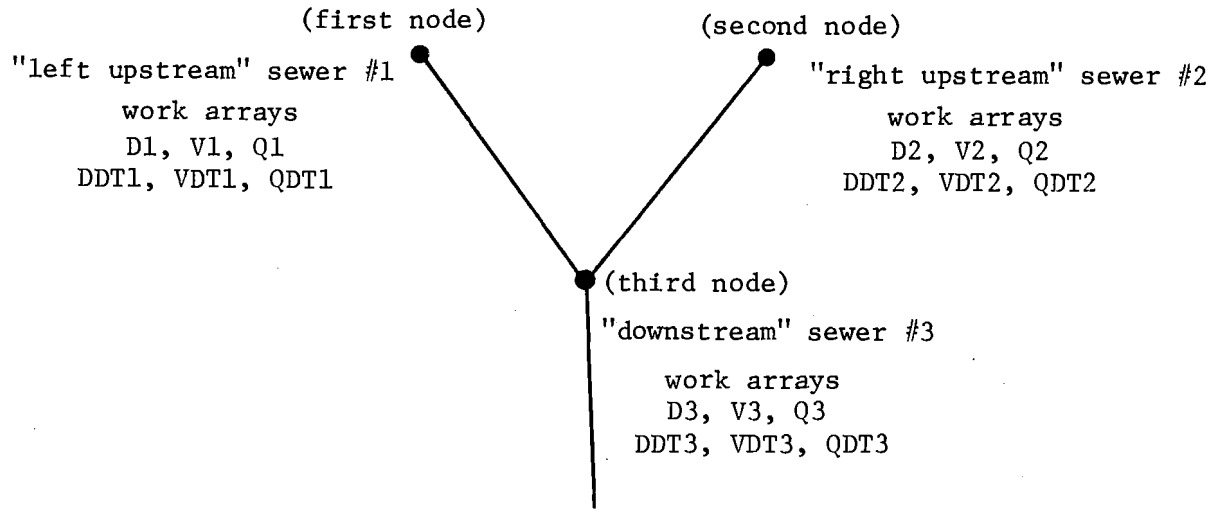


FIG. 16. SEWER NUMBERING OF Y-SEGMENT FOR EVALU8R SUBROUTINE

Upon the completion of computations at time step $t + \Delta t$, the contents of the pseudoregisters pointing to the arrays QDTn, DDTn, and VDTn are swapped with those pointing to the arrays Qn, Dn, and Vn, respectively. Thus, instead of copying the entire nine "new values" arrays over into the nine "current values" arrays, this name interchange operation makes the "new values" arrays the "current values" arrays and vice-versa, which is a more efficient technique.

6.4.2. Management of FLOWBLOCK

The most sophisticated data management technique for controlled variables in the program is for the management of FLOWBLOCKS. The FLOWBLOCK arrays, which store the discharge hydrographs at sewer entrances, must have not only variable upper and lower bounds, like a normal controlled array, but also readily available generation on short notice, like a standard based array.

As discussed in Subsection 6.2.2, in the program, FLOWBLOCKS are allocated as controlled arrays. After their allocation, the contents of the pseudoregister associated with the FLOWBLOCK controlled variable is saved in the NODE unit with which the particular FLOWBLOCK is affiliated. The pseudoregister is also used to update the LASTALLOC pointer which always points to the real "top" of the controlled variable stack. Thus, the EVALU8R subroutine, when referring to a FLOWBLOCK value for the FLOWBLOCK afiliated with a particular NODE, needs only to use its NODE pointer to recover the proper pseudoregister contents. The NODE pointer is stuffed into the FLOWBLOCK pseudoregister, and reference to the desired element can thus be made. When FLOWBLOCKS for more than one NODE must be accessed simultaneously, two dummy controlled variables TFB and TFB2 can be used by stuffing the pointers into their pseudoregisters instead. Thus, all three FLOWBLOCKS for a Y-segment can be accessed simultaneously.

6.4.3. Sequencing Through Y-Segments of Network

As mentioned at the beginning of this chapter, the Y-segments of a sewer network is sequenced following a well defined order by SRCHTREE. In order to determine the proper sequence, a pointer to the root node of the network is passed as a parameter to SRCHTREE. The SRCHTREE subroutine then examines the Y-segment, the downstream node of which is pointed by the parameter pointer. If the inflow at the first upstream node (Fig. 15) of the Y-segment being examined is not known, SRCHTREE invokes itself recurrsively passing as a parameter the pointer to the junction node of the Y-segment. This recurrsive process effectively reduces the size of the network to that part upstream of the junction of the current Y-segment being considered.

In the process when the first inflow of a Y-segment is found known, SRCHTREE performs a similar recurrsive call along the second sewer until it finds a Y-segment with the inflows at both of its upstream nodes are known. Once such a Y-segment is found, the EVALU8R subroutine is called by SRCHTREE to perform computation of the flows within the segment. The computed inflow hydrograph into the downstream sewer is the FLOWBLOCK for the junction node of the segment. EVALU8R cannot be called for a Y-segment if one or both of its inflow hydrographs is unknown. After the junction node FLOWBLOCK is established, the control is passed back from EVALU8R to SRCHTREE. The SRCHTREE subroutine then returns to invoke the recurrsive search procedure for subsequent Y-segments starting again from the root node of the network.

7. COMPUTER PROGRAM DISTRIBUTION TAPE

7.1. Distribution Tapes

As mentioned in Chapter 4, the program of the ISS Model can be implemented through the use of either a distribution tape or the program listing. The latter will require the assistance of an experienced programmer and may cause unnecessary computer expenses. Therefore, the use of the program distribution tape is recommended. The distribution tape can be made available to all prospective users for the cost of the magnetic tape, handling, and computer time required for duplication.

Standard distribution tapes are provided on 1600 bytes per inch, 2400 ft magnetic tapes. The following information should be supplied by the user when requesting for a distribution tape:

a) Desired tape density, 800 or 1600 bpi.

b) Model of computer system, e.g., IBM System 370, model 168 MP.

c) Operating system in use, e.g., OS/MVT with HASP.

d) Region size available for the program, e.g., 220K.

e) Type of disk or other direct-access device to which the program will be transferred from the distribution tape, e.g., 2314, 2319, 3330.

f) Available compilers at the installation, e.g., either, neither or both of the OS PL/1 (F) and PL/1 Optimizing compilers.

g) Computer programming experience of the user, e.g., FORTRAN IV, PL/1.

7.2. Operational Procedure

Implementation of the computer program of the ISS Model through the use of distribution tape consists of installation of the distribution tape and execution of the program thus installed.

7.2.1. Program Installation

The distribution tape is largely self installing and its installation into a compatible computer system consists of the following operations which are usually performed by a computer operator.

a) Read into the user's computer system about ten JCL (Job Control Language) cards for the installation.

b) Read the distribution tape into the user's computer system. The contents of the tape will be transferred into a disk or other direct-access device.

c) After the tape is read in, the computer will punch out a deck of cards which constitute the JCL cards to execute the ISS Model program.

This program installation operation needs to be performed only once so long as the program is kept in the disk or other direct-access device and the user keeps the program execution JCL card deck. In Step (a) above, the exact number and contents of the installation JCL cards vary depending on the hardware and software configuration of the user's computer system as partially reflected by items (a) through (g) listed in the preceding section. The instruction sheet which accompanies the distribution tape will include either the complete listing of the JCL cards or the necessary information to prepare them.

7.2.2. Program Execution

After the ISS Model computer program has been installed as discussed in the preceding subsection, the execution of the program for design or operational studies of sewer network systems consists of the following five steps.

a) Prepare the data cards following the format discussed in Sec. 5.1.

b) Arrange the data cards in proper order and attach them after the execution JCL card deck such as that shown in Fig. 17 which will further be discussed in the later part of this subsection.

c) Read in the properly arranged JCL and data card deck to the computer system. This input will automatically be verified. If no error is found, the mathematical simulation will then be proceeded.

d) The user will receive from the computer the print-out of the properties of the networks and inflow hydrographs as illustrated in Fig. 10 and discussed in Subsection 5.2.1 , and the print-out of the results of the numerical simulation as illustrated in Fig. 11 and discussed in Subsection 5.2.2. The user may also receive plots of discharge and depth graphs at the entrance of the sewers if the plotting program is used.

e) The user then reclaims his cards and he should save the JCL cards for future analysis of other networks, if so expected.

It may be appropriate to mention here that several network design and operational analyses can be made in a single run of the ISS Model. In arranging the order of cards for Step (b), the execution JCL card deck should always come first, followed by the START card, data cards, and CEND card for the first network. The START card, data cards, and CEND card for the second network follow immediately after the CEND card of the first network. The cards for the third and following networks are arranged in the same manner. This complete deck of cards for all the networks is read simultaneously. The program then executes the networks one at a time, following the order in which their input decks are read in.

As to the order of the input data cards for each network, the following are the general rules, which should be followed:

(i) The data cards for each network must start with a START card.

(ii) The user assigned node numbers must appear on the NODE cards before they are referred to in any other control or data cards. The safe procedure is to put the NODE cards immediately after the START card as shown in Fig. 17.

(iii) Each inflow hydrograph must be specified as a unit on a set of cards consisting of a FBD (or JFBD) card, pairs of time-discharge values punched on one or more cards, and a FEND (or JEND) card, in that order, as discussed in Subsection 5.1.3. The order of different hydrographs, however, is immaterial.

(iv) The data deck for each network must be terminated by a CEND card.

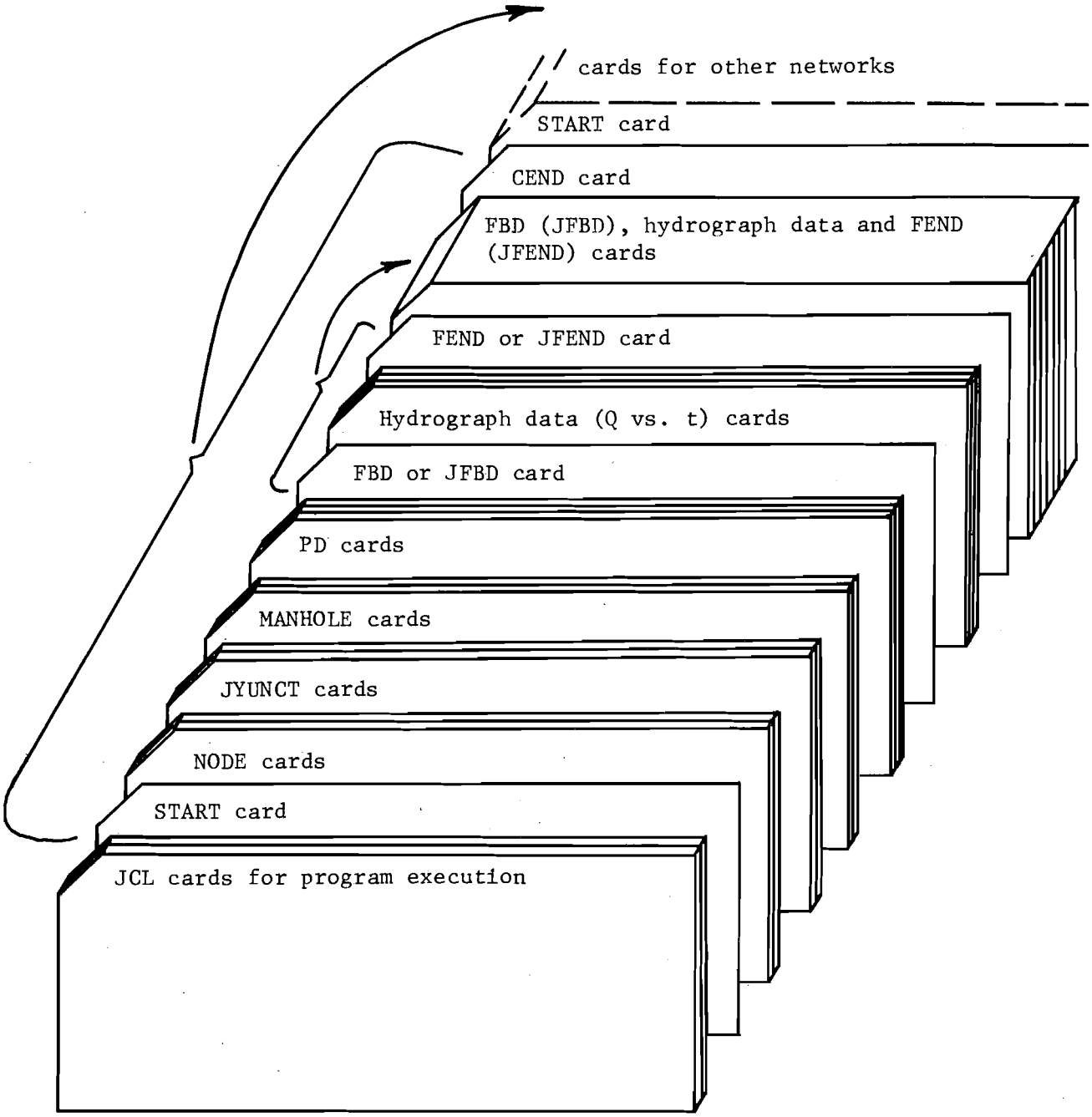


FIG. 17. RECOMMENDED ARRANGEMENT OF INPUT CARDS

REFERENCES

1. Chow, V. T., "Open-Channel Hydraulics", McGraw-Hill Book Co., Inc., New York, pp. 262-265 and 525-528, 1959.

2. Sevuk, A. S., "Unsteady Flow in Sewer Networks", Ph.D. thesis, Dept. of Civil Eng., Univ. of Illinois at Urbana-Champaign, 142 p., 1973.

3. Strahler, A. N., "Quantitative Geomorhology of Drainage Basins and Channel Networks", Handbook of Applied Hydrology, ed. by V. T. Chow, pp. 4.39-4.76, McGraw-Hill Book Co., Inc., New York, 1964.

4. Streeter, V. L., and E. B. Wylie, "Hydraulic Transients", McGraw-Hill Book Co., Inc., New York, pp. 239-259, 1967.

5. Yen, B. C., "Methodologies for Flow Prediction in Urbana Storm Drainage Systems", Water Resources Center Research Rept. No. 72, Univ. of Illinois at Urbana-Champaign, 150 p., August 1973.

APPENDIX A

Program Block Diagram and List of Subroutines

An overall block diagram for the computer program is given in Fig. A1 indicating composition of major program subroutines and in Fig. A2 of modules. Parts of the program are compiled separately and then linkage edited together with the assembler language routine. The separate compilation of various PL/1 sections was felt desirable due to their disparate nature, and in the name of modular program structure while retaining efficiency. The separate compilations shown schematically in Fig. A2 are listed together with their attribute and cross reference tables in Appendix B. The Subroutines shown in the block diagram together with their functions are listed in the following in alphabetical order.

Subroutine	Function
BADNODE	scans conflict type errors among node attributes and calls ERRMES to print error or warning messages.
CIRCLE	computes geometrical parameters of circular sewers for given diameter and flow depth.
CODEGEN	generates interpretable subfunction operations.
COEFFICIENT	simplifies characteristic equations at junction stations (see Sec. 3.1).
COMPILE	is an expression compiler, calls EXPR and TERM to verify and compile the flow equation specified on the OUTLET-CONTROL card (see Subsection 5.1.2).
DCOMPIL	is a compiler cleanup routine, frees all the storages used by COMPILE at the end of execution for each sewer network.
DCRIT	computes critical flow depth for specified discharge.
DIAMSET	is a design routine used together with the subroutine FINDIAM; it computes pipe size for specified or estimated rate of peak flow (see Sec. 2.4).

Subroutine	Function
DNORMAL	computes steady uniform flow depth for specified discharge.
DNSTRM	computes flow conditions at downstream boundary station (see Sec. 3.4).
EMSG	prints error messages via ERRMES.
ERRMES	prints error messages passed as parameters.
EVALU8R	performs numerical computations over Y-segment by calling various subroutines.
EXPR	evaluates an expression by invoking EXPR and TERM.
FINDIAM	is a design routine used together with the subroutine DIAMSET; it gives the commercial pipe size according to the size computed by DIAMSET (see Sec. 2.4).
GETI8PR GETF8PR GETTPR GETTGPR	are a group of four entries in assembler language routine; they return addresses of pseudoregisters pointing to assorted controlled variables (see Sec. 6.3).
GRAPHER	plots discharge hydrographs at sewer entrances with line printer.
INCOND	computes initial steady state water surface profiles (see Sec. 2.3).
INTER	performs computations at interior stations (see Sec. 3.2).
JUNCTION	performs computations at junction stations (see Sec. 3.5).
JUNC DROP	computes critical flow condition at drop structures.
LASTPIPE	is the post processing routine; deallocates final FLOWBLOCK storage.
MAKE-TABLE	is a pipe summary table output director, produces the pipe summary table calling PIPE-PRINT and PIPE-HEADINGS (see Figs. 10 and 11).
NETWORK	is the main driver, a call program.
PLOTTER	provides hooks for data capture for subsequent incremental plotting (see Fig. 12).
PLUCK	dynamically rearranges PL/1 controlled array stacks (see Sec. 6.3).
PRTCHG	prints computed hydrograph and depth graphs (see Fig. 11).

Subroutine	Function
PRTFLO	prints values of discharge, velocity, and depth along each sewer, at computational grid points, at regular time intervals (see Fig. 1).
PRTIHG	prints specified inflow hydrographs (see Fig. 11).
SETUP	is the input deck scanner and data structure constructor (see Sec. 6.1).
SRCHTREE	is the recurrsive sequencing director, determines the appropriate sequence of overlapping Y-segments and invokes EVALU8R to perform the numerical computations (see Subsection 6.4.3).
TERM	evaluates a single term in flow equation of control structure by invoking TERM and EXPR.
UPSTRM	performs computations at upstream boundary stations (see Sec. 3.3).
VALUATE	evaluates the flow equation output from COMPILE.

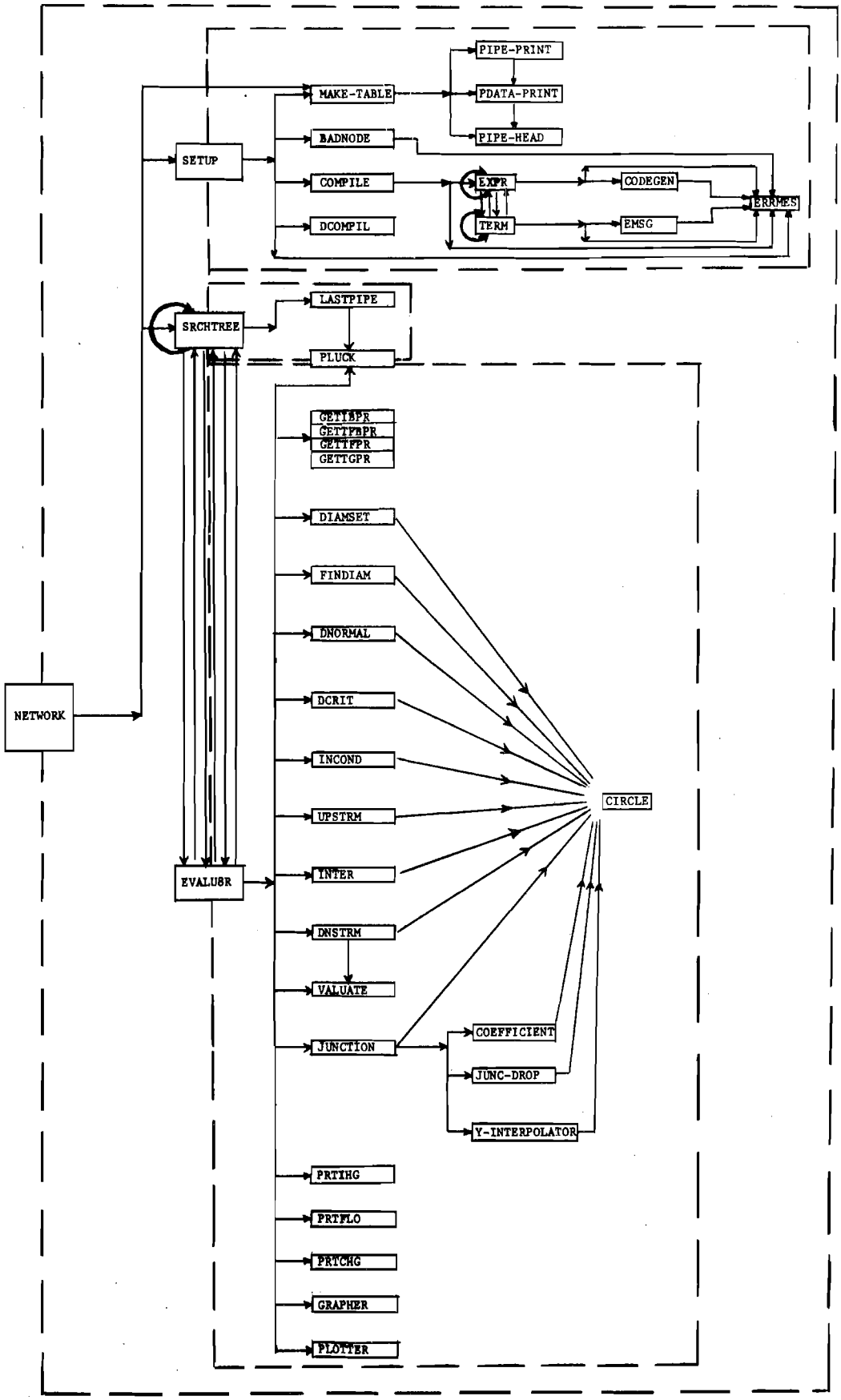


FIG. A1. COMPOSITION OF MAJOR PROGRAM SUBROUTINES

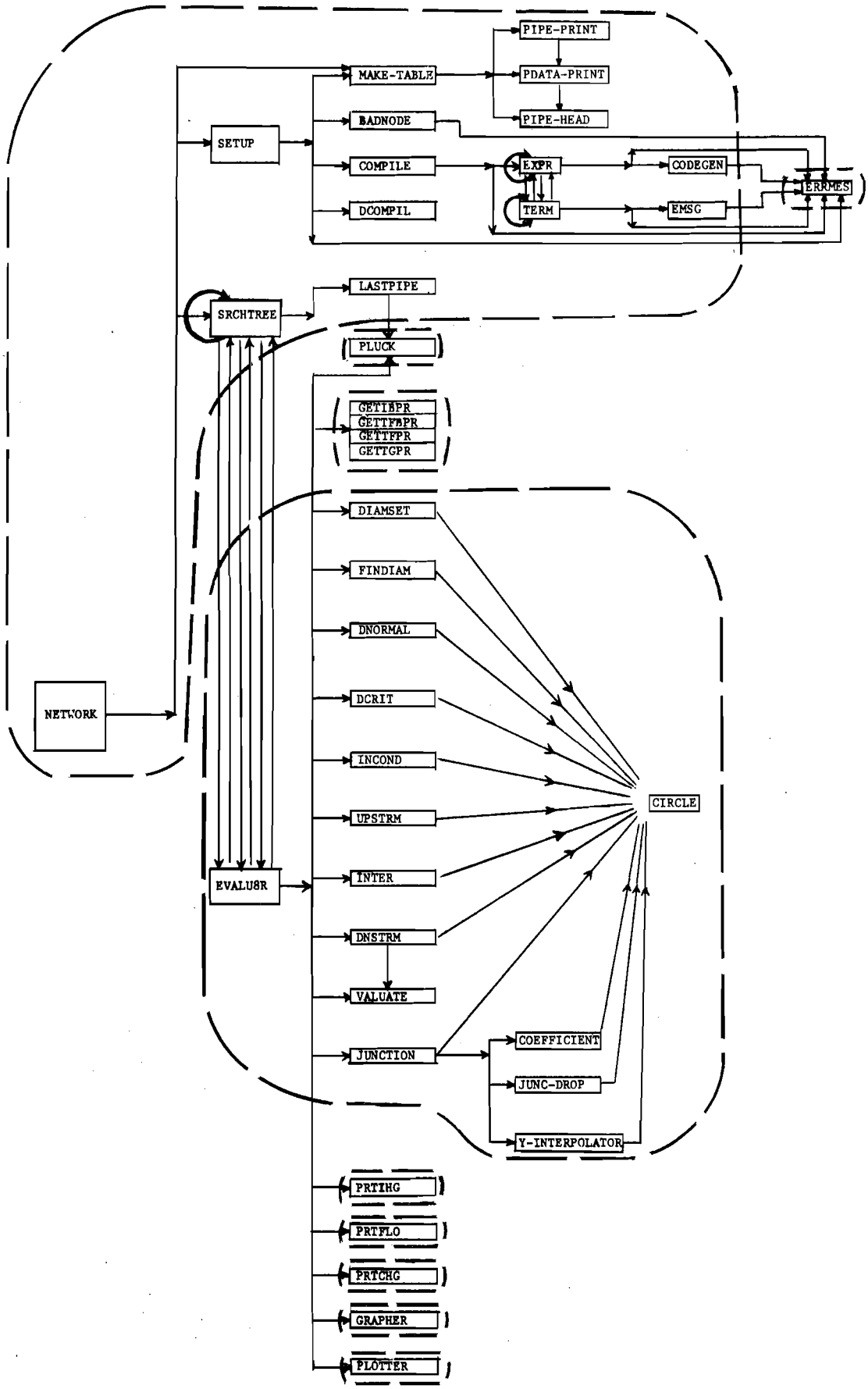


FIG. A2. MODULAR STRUCTURE OF PROGRAM

APPENDIX B

Program Listing

```
PL/I OPTIMIZING COMPILER          NETWORK:  PROC OPTIONS(MAIN) REORDER;

                        SOURCE LISTING

   STMT LEV NT

     1      0  |NETWORK:  PROC OPTIONS(MAIN) REORDER;                                        |00000100
               |/*  MAIN DRIVER -- THIS IS MAINLY JUST A CALLING PROGRAM. */               |00000200
     2    1 0  |DECLARE                                                                     |00000300
               |1 NODE BASED(NODE_PTR),                                                     |00000400
               |2 NODE_# FIXED BIN(16),                                                     |00000500
               |2 DOWNSTREAM_NODEPTR POINTER,          /* NULL IF ROOT NODE. */             |00000600
               |2 UPSTREAM_NODEPTR(2) POINTER,          /* BOTH NULL IF INPUT */            |00000700
               |2 DOWNSTREAM_PIPE_INFO,                                                     |00000800
               |3 PIPE_LENGTH FIXED BIN(31),                                                |00000900
               |3 PIPE_SLOPE FLOAT BIN,                                                     |00001000
               |3 PIPE_DIAMETER FLOAT BIN,                                                  |00001100
               |3 PIPE_DROP FLOAT BIN,                                                      |00001200
               |3 PIPE_ROUGHNESS FLOAT BIN,                                                 |00001300
               |3 FUNCTION_INFO,                                                            |00001310
               |4 FUNCPTR POINTER,                                                          |00001320
               |4 FUNCVARA FIXED BIN(15),                                                   |00001330
               |4 FUNCVARR FIXED BIN(15),                                                   |00001340
               |2 NODE_TYPE,                                                                |00001400
               |3 MANHOLE BIT(1) ALIGNED,                                                   |00001500
               |3 Y_JUNCTION BIT(1) ALIGNED,                                                |00001600
               |3 INPUT_STATION BIT(1) ALIGNED,                                             |00001700
               |3 ROOT_STATION BIT(1) ALIGNED,                                              |00001800
               |3 IMAGINARY BIT(1) ALIGNED,                                                 |00001900
               |3 CRASHED BIT(1) ALIGNED,                                                   |00001910
               |2 FLOWBLOCK_POINTER POINTER,                                                |00002000
               |2 YBLOCK_POINTER POINTER,                                                   |00002100
               |2 CRITICAL_DEPTH FLOAT BIN,                                                 |00002200
               |2 TABLE_PAGE# FIXED DEC(4),                                                 |00002300
               |2 DC FLOAT BIN,                                                             |00002400
               |2 DNORM FLOAT BIN,                                                          |00002500
               |2 MANHOLAREA FLOAT,                                                         |00002600
               |PLOCK ENTRY(POINTER,POINTER),                                               |00002700
               |ERRMES ENTRY(CHAR(*)) EXTERNAL,                                             |00002800
               |FLOWBLOCK(*) CONTROLLED EXTERNAL FLOAT BIN,                                 |00002900
               |EVALUBR ENTRY(POINTER,POINTER,POINTER) EXTERNAL,                            |00003000
               |MAXNODES FIXED BIN(15) INIT(200),                                           |00003400
               |NET# /* COUNTER FOR # OF TREES EVALUATED. */ FIXED BIN INIT(0),            |00003500
               |ROOT_NODE_# FIXED BIN(15),                                                  |00003700
               |#_NODES_ALLOCATED FIXED BIN(15) INIT(0),                                    |00003800
               |#_NODES_LEFT FIXED BIN(15),                                                 |00003900
               |(TPTR1,TPTR2,TPTR3) POINTER,                                                |00004000
               |ROOT_POINTER POINTER,                                                       |00004100
               |/*                                                                          |00004200
               |INITCOND ENTRY(POINTER,FLOAT BIN,FLOAT BIN,FLOAT BIN, FLOAT BIN,           |00004300
               |FLOAT BIN),                                                                 |00004400
               |*/                                                                          |00004500
               |NODE_LIST(#_NODES_ALLOCATED) POINTER CONTROLLED EXTERNAL,                   |00004600
               |PAGENO FIXED DEC(4) INIT(1) EXTERNAL,                                       |00004700
               |PAGEBIT BIT(1) INIT('1'B),                                                  |00004800
               |DEBUG CHAR(20) VAR EXTERNAL,                                                |00004900
               |SOPARM CHAR(30) VAR EXTERNAL,                                               |00005000
               |GRAPHFL BIT(1) EXTERNAL,                                                    |00005100
               |SETDIAM BIT(1) EXTERNAL,                                                    |00005200
               |TODAY CHAR(8),                                                              |00005300
               |LASTALLOC POINTER EXTERNAL,PSEUDOREGISTER POINTER BASED(PSEUDOBASE),        |00005400
               |(GETFBPR,GET18PR,GETTPR,GETTGPR) ENTRY(POINTER),                            |00005550
               |(NULL,DATE,ONSOURCE) BUILTIN;                                               |00005600

     3    1 0  |TODAY=DATE;                                                                 |00005700
     4    1 0  |TODAY=SUBSTR(TODAY,3,2)||'/'||SUBSTR(TODAY,5,2)||'/'||SUBSTR(TODAY,1,2)    |00005800
               |;                                                                           |
     5    1 0  |CALL GETFBPR(ADDR(PSEUDOBASE));                                             |00005900
               |                                                                            |00006100
     6    1 0  |ON ENDPAGE(SYSPRINT) BEGIN;                                                 |00006300
     7    2 0  |IF PAGEBIT THEN PAGEBIT='0'B;  ELSE PUT PAGE;                               |00006400
     8    2 0  |PUT FILE(SYSPRINT) EDIT('DATE '||TODAY,                                     |00006500
               |'UNIVERSITY OF ILLINOIS CIVIL ENGINEERING--STORM SEWER SIMULATION PROGR     |00006600
               |AM',                                                                        |
               |'PAGE ',PAGENO)                                                             |00006700
               |(A,COL(25),A,COL(110),A,F(4));                                              |00006800
    10    2 0  |PAGENO=PAGENO+1;                                                            |00006900
    11    2 0  |PUT SKIP(3);                                                                |00007000
    12    2 0  |END;                                                                        |00007100
               |                                                                            |00007200
```

```
13   1  0  |BUILD_THE_TREE:                                                    |00007300
           |DEBUG='';                                                         |00007400
14   1  0  |SOPARM='';                                                        |00007500
15   1  0  |LASTALLOC=NULL;                                                   |00007600

16   1  0  |CALL SETUP;                                                       |00007700
           |/* TREE IS NOW CONSTRUCTED.  NOW TO EVALUATE IT. */               |00007800
17   1  0  |CALL SRCHTREE(ROOT_POINTER);  /* THIS DOES ALL FLOW EVALUATION. */|00007900
18   1  0  |CALL MAKE_TABLE(ROOT_POINTER);                                    |00008000
           |/* NEXT FREE THE TREE, AND GO GET THE NEXT CONTROL CARDS. */      |00008100
19   1  0  |DO I=1 TO HBOUND(NODE_LIST,1);                                    |00008200
20   1  1  |TPTR1=NODE_LIST(I);                                               |00008300
21   1  1  |FREE TPTR1->NODE;                                                 |00008400
22   1  1  |END;                                                              |00008500
23   1  0  |FREE NODE_LIST;                                                   |00008550
           |/* THE TREE NETWORK IS NOW ALL GONE. */                          |00008600
24   1  0  |DO WHILE(ALLOCATION(FLOWBLOCK));                                  |00008700
25   1  1  |FREE FLOWBLOCK;                                                   |00008800
26   1  1  |END;                                                              |00008900
27   1  0  |GOTO BUILD_THE_TREE;                                              |00009000



28   1  0  |SRCHTREE:  PROC(ROOT_POINTER) RECURSIVE;                          |00009100
           |/* THIS IS THE SUBROUTINE WHICH RECURSIVELY SEQUENCES THROUGH THE    */|00009110
           |/* PIPE NETWORK "TREE".  WHEN IT FINDS AN EVALUATEABLE PIPE TRIPLET, */|00009120
           |/* IT INVOKES THE EVALUOR SUBSYSTEM.                                */|00009130
29   2  0  |DCL I FIXED BIN(31), TEMPPTR POINTER, ROOT_POINTER POINTER, NULL BUILTI|00009200
           |N,                                                                |00009300
           |(NODE1_PTR, NODE2_PTR, JNODE_PTR) POINTER;                        |00009400
30   2  0  |NODE1_PTR, NODE2_PTR, JNODE_PTR=NULL;                             |00009500
31   2  0  |DO I=1 TO 2;                                                      |00009600
32   2  1  |TEMPPTR=ROOT_POINTER->UPSTREAM_NODEPTR(I);                        |00009700
           |/* TEMPPTR NOW POINTS TO ONE "UPSTREAM" NODE. */                  |00009800
33   2  1  |IF TEMPPTR¬=NULL THEN IF TEMPPTR->NODE.FLOWBLOCK_POINTER=NULL     |00009900
           |/* I.E. IF THERE A) IS THIS UPSTREAM NODE, AND&B) IT HAS NO FLOWBLOCK */|00010000
           |THEN DO;                                                          |00010100
34   2  2  |CALL SRCHTREE(TEMPPTR);                                           |00010200
           |/* TAKE THE PRECEEDING NODE(POINTED TO BY TEMPPTR) AS THE "ROOT" NODE*/|00010300
35   2  2  |IF I=1 THEN NODE1_PTR=TEMPPTR;                                    |00010400
36   2  2  |ELSE NODE2_PTR=TEMPPTR;                                           |00010500
37   2  2  |END;                                                              |00010600
38   2  1  |ELSE /* NODE EXISTS, AND WE KNOW THE FLOW THERE. */               |00010700
           |IF I=1 THEN NODE1_PTR=TEMPPTR;                                    |00010800
39   2  1  |ELSE NODE2_PTR=TEMPPTR;                                           |00010900
40   2  1  |ELSE /*NO I-TH UPSTREAM NODE FROM "ROOT"NODE. */                  |00011000
           |IF ¬ROOT_POINTER->NODE_TYPE.ROOT_STATION THEN SIGNAL ERROR;       |00011100
41   2  1  |ELSE DO; /* ROOT_POINTER POINTS TO THE REAL ROOT NODE.  THIS MEANS  */|00011200
           |/* THAT WE KNOW THE FLOW INTO THE BOTTOMMOST PIPE IN THE TREE.  FIRST*/|00011300
           |/* THOUGH IT MIGHT BE A GOOD IDEA TO MAKE SURE. */               |00011400
42   2  2  |IF I=1 THEN SIGNAL ERROR;                                         |00011500
43   2  2  |TEMPPTR=ROOT_POINTER->UPSTREAM_NODEPTR(1);                        |00011600
44   2  2  |CALL LASTPIPE(TEMPPTR);                                           |00011700
45   2  2  |RETURN;                                                           |00011800
46   2  2  |END;                                                              |00011900
47   2  1  |END;                                                              |00012000
48   2  0  |JNODE_PTR=ROOT_POINTER;                                           |00012100
49   2  0  |CALL EVALUOR(NODE1_PTR,NODE2_PTR,JNODE_PTR);                      |00012200
50   2  0  |END SRCHTREE;                                                     |00012300



51   1  0  |BADNODE:  PROC(NODEPTR) RETURNS(BIT(1));                          |00013000
           |/* THIS IS THE SUBROUTINE WHICH CHECKS FOR "CONFLICT" TYPE ERRORS    */|00013010
           |/* DURING THE READING OF THE NETWORK DATA CARDS.  UPON FINDING SUCH  */|00013020
           |/* AN ERROR, "ERRMES" IS CALLED TO PRINT AN ERROR MESSAGE, AND A 1   */|00013030
           |/* IS RETURNED.  IF EVERYTHING LOOKS OK SO FAR, A ZERO IS RETURNED.  */|00013040
52   2  0  |DCL NODEPTR POINTER,                                              |00013100
           |I FIXED BIN(15) INIT(0),                                          |00013200
           |HOLDING CHAR(95) VAR;                                             |00013300
53   2  0  |I=(NODEPTR->MANHOLE='1'B)+(NODEPTR->Y_JUNCTION='1'B)              |00013400
           |+(NODEPTR->INPUT_STATION='1'B)+(NODEPTR->ROOT_STATION='1'B)       |00013500
           |+(NODEPTR->IMAGINARY='1'B);                                       |00013600
54   2  0  |IF I>1 THEN DO;                                                   |00013700
55   2  1  |HOLDING='YOU ARE TRYING TO DEFINE THE LAST NODE AS BOTH ';        |00013800
56   2  1  |IF NODEPTR->MANHOLE THEN HOLDING=HOLDING||'A MANHOLE AND ';       |00013900
57   2  1  |IF NODEPTR->Y_JUNCTION THEN HOLDING=HOLDING||'A Y-JUNCTION AND '; |00014000
58   2  1  |IF NODEPTR->INPUT_STATION THEN HOLDING=HOLDING||'AN INPUT STATION AND '|00014100
           |;                                                                |00014200
59   2  1  |IF NODEPTR->IMAGINARY THEN HOLDING=HOLDING||'AN IMAGINARY NODE AND '; |00014300
60   2  1  |IF NODEPTR->ROOT_STATION THEN HOLDING=HOLDING||'A ROOT STATION AND '; |00014400
61   2  1  |HOLDING=SUBSTR(HOLDING,1,LENGTH(HOLDING)-5);                      |00014500
62   2  1  |CALL ERRMES(HOLDING);                                             |00014600
63   2  1  |RETURN('1'B);                                                     |00014700
64   2  1  |END;                                                              |00014800
           |/* IF WE GET HERE, THE NODE HAS NOT BEEN "DOUBLY DEFINED".  NEXT,  */|00014900
           |/* CHECK IF SOME OTHER "BAD NODE" CRITERIA ARE SPECIFIED. */       |00015000
```

```
65   2  0  |IF NODEPTR->ROOT_STATION THEN IF NODEPTR->UPSTREAM_NODEPTR(2)¬=NULL     |00015100
           |THEN DO;                                                                |00015200
66   2  1  |CALL ERRMES('THE ROOT STATION CANNOT HAVE TWO INCOMING PIPES');          |00015300
67   2  1  |RETURN('1'B);                                                           |00015400
68   2  1  |END;                                                                    |00015500
69   2  0  |ELSE DO;                                                                |00015600
70   2  1  |IF NODEPTR->DOWNSTREAM_NODEPTR¬=NULL THEN DO;                           |00015700
71   2  2  |CALL ERRMES('THE ROOT STATION CANNOT HAVE AN OUTGOING PIPE');           |00015800
72   2  2  |RETURN('1'B);                                                           |00015900
73   2  2  |END;                                                                    |00016000
74   2  1  |IF NODEPTR->FLOWBLOCK_POINTER¬=NULL THEN DO;                            |00016100
75   2  2  |CALL ERRMES('YOU HAVE SPECIFIED THE INPUT TO THE ROOT STATION');        |00016200
76   2  2  |RETURN('1'B);                                                           |00016300
77   2  2  |END;                                                                    |00016400
78   2  1  |END;                                                                    |00016500
79   2  0  |IF NODEPTR->INPUT_STATION THEN IF NODEPTR->UPSTREAM_NODEPTR(1)¬=NULL    |00016600
           |THEN DO;                                                                |00016700
80   2  1  |CALL ERRMES('AN INPUT STATION IS NOT ALLOWED TO HAVE AN UPSTREAM PIPE') |00016800
           |;                                                                       |00016900
81   2  1  |RETURN('1'B);                                                           |00017000
82   2  1  |END;                                                                    |00017100
83   2  0  |IF NODEPTR->IMAGINARY THEN IF NODEPTR->UPSTREAM_NODEPTR(1)¬=NULL        |00017200
           |THEN DO;                                                                |00017300
84   2  1  |CALL ERRMES('AN IMAGINARY NODE IS NOT ALLOWED TO HAVE AN UPSTREAM PIPE'|00017400
           |);                                                                      |00017500
85   2  1  |RETURN('1'B);                                                           |00017600
86   2  1  |END;                                                                    |00017700
87   2  0  |IF NODEPTR->YBLOCK_POINTER¬=NULL THEN IF NODEPTR->IMAGINARY |          |00017800
           |NODEPTR->INPUT_STATION | NODEPTR->ROOT_STATION THEN DO;                 |00017900
88   2  1  |CALL ERRMES('A JFBD CARD IS ONLY PERMITTED FOR MANHOLE AND Y-JUNCTION N|00018000
89   2  1  |ODES'); RETURN('0'B); END;                                              |00018100
91   2  0  |RETURN('0'B);                                                           |00018200
92   2  0  |END BADNODE;                                                            |00018300
```

```
93   1  0  |SETUP: PROC;                                                            |00018400
           |/* THIS IS THE SUBROUTINE WHICH READS IN ALL INPUT CARDS AND        */|00018410
           |/* CONSTRUCTS THE NETWORK "TREE". WHEN IT RETURNS, THE TREE IS      */|00018420
           |/* BUILT, VERIFIED, AND READY TO BE PROCESSED.                      */|00018430
94   2  0  |DCL BUFFER CHAR(80) BASED(BUFFTR), BUFFTR POINTER,                      |00018500
           |(TI,TL) FLOAT BIN STATIC EXTERNAL,                                      |00018600
           |1 OUTVARS STATIC EXTERNAL,                                              |00018610
           |(2 FT, 2 FDI, 2 FV, 2 FDE) FLOAT BIN,                                  |00018620
           |2 FUNCPTR POINTER,                                                      |00018630
           |2 FUNCVARA FIXED BIN(15),                                               |00018640
           |2 FUNCVARR FIXED BIN(15),                                               |00018650
           |COMPILE ENTRY(CHAR(*), FIXED BIN(15), FIXED BIN(15), ) EXTERNAL         |00018660
           |RETURNS(POINTER),                                                       |00018670
           |DCOMPIL ENTRY EXTERNAL,                                                 |00018680
           |COMPTST BIT(1) EXTERNAL, /* IF TURNED ON, GENERATES DEBUG OUTPUT */     |00018690
           |MSGBUF CHAR(100) VAR,                                                   |00018700
           |SYSIN FILE RECORD SEQUENTIAL INPUT ENV(RECSIZE(80) CONSECUTIVE TOTAL),  |00018710
           |EXECUTE BIT(1);                                                         |00018800

95   2  0  |ON ENDFILE(SYSIN) BEGIN;                                                |00018900
96   3  0  |SIGNAL ENDPAGE(SYSPRINT);                                               |00019000
97   3  0  |PUT EDIT('*ALL CONTROL CARDS HAVE BEEN PROCESSED.',                     |00019100
           |NET#,' NETWORKS HAVE BEEN PROCESSED.  EXECUTION TERMINATES.')           |00019200
           |(SKIP(5),A,F(4),A);                                                     |00019300
98   3  0  |GOTO END_MAIN;                                                          |00019400
99   3  0  |END;                                                                    |00019500

100  2  0  |ON ERROR BEGIN;                                                         |00019510
101  3  0  |IF SUBSTR(BUFFER,1,6)='START ' THEN READ FILE(SYSIN) SET(BUFFTR);       |00019520
102  3  0  |CALL ERRMES('ERROR ENCOUNTERED WHILE PROCESSING "'||SUBSTR(BUFFER,1,    |00019530
           |INDEX(BUFFER,' ')-1)||'" CARD');                                        |00019540
103  3  0  |GOTO FLUSH;                                                             |00019550
104  3  0  |END;                                                                    |00019560

105  2  0  |START_READING_CONTROL_CARDS:                                            |00019600
           |READ FILE(SYSIN) SET(BUFFTR);                                           |00019700
106  2  0  |GOT_FIRST_CONTROL_CARD:                                                 |00019750
           |IF SUBSTR(BUFFER,1,72)=' ' THEN GOTO START_READING_CONTROL_CARDS;       |00019800
107  2  0  |MAXNODES=200;  /* DEFAULT MAX. # OF NODES. */                           |00019900
108  2  0  |TL=250;  /* DEFAULT TIME LIMIT ON ITERATIONS IN SECONDS. */            |00020000
109  2  0  |TI=30;  /* DEFAULT TIME INCREMENT IN SECONDS. */                       |00020100
110  2  0  |OUTVARS.FUNCPTR=NULL; /* SET POINTER TO COMPILED OUTPUT CTRL FUNCTION*/|00020110
111  2  0  |OUTVARS.FUNCVARA, OUTVARS.FUNCVARR = 0;                                 |00020120
112  2  0  |CALL DCOMPIL;  /* FREE ANY EXISTING COMPILED STATEMENTS. */             |00020130
113  2  0  |EXECUTE='1'B;                                                           |00020200
114  2  0  |SIGNAL ENDPAGE(SYSPRINT);                                               |00020300
115  2  0  |NET#=NET#+1;                                                            |00020400
```

```
115   2  0  |PUT FILE(SYSPRINT) EDIT('CONTROL CARD LISTING FOR STORM SEWER NETWORK #|00020500
           |',NET#,BUFFER)(SKIP(2),X(35),A,F(2),SKIP(3),X(20),A);                  |00020600
117   2  0  |SUBSTR(BUFFER,73)=';';                                                 |00020700
118   2  0  |IF SUBSTR(BUFFER,1,6)¬='START ' THEN DO;                               |00020800
119   2  1  |CALL ERRMES('MISSING "START" CARD');                                   |00020900
120   2  1  |FLUSH:  DO WHILE(SUBSTR(BUFFER,1,5)¬='CEND ');                         |00021000
121   2  2  |IF SUBSTR(BUFFER,1,6)='START ' THEN GOTO GOT_FIRST_CONTROL_CARD;       |00021010
122   2  2  |READ FILE(SYSIN) SET(BUFPTR);                                          |00021100
123   2  2  |END;                                                                   |00021200
124   2  1  |GOTO START_READING_CONTROL_CARDS;                                      |00021300
125   2  1  |END;                                                                   |00021400
126   2  0  |ELSE IF SUBSTR(BUFFER,7,66)¬=' ' THEN DO;                              |00021500

127   2  1  |I=INDEX(BUFFER,'NOEXEC');                                              |00021600
128   2  1  |IF I>0 THEN DO;                                                        |00021700
129   2  2  |EXECUTE='0'B;                                                          |00021800
130   2  2  |SUBSTR(BUFFER,I,6)='      ';                                          |00021900
131   2  2  |END;                                                                   |00022000

132   2  1  |I=INDEX(BUFFER,'DESIGN');                                              |00022100
133   2  1  |SETDIAM='0'B;                                                          |00022200
134   2  1  |IF I>0 THEN DO;  SETDIAM='1'B;  SUBSTR(BUFFER,I,6)='      ';  END;    |00022300

135   2  1  |I=INDEX(BUFFER,'NOGRAPH');                                             |00022400
139   2  1  |GRAPHPL='1'B;                                                          |00022500
140   2  1  |IF I>0 THEN DO;                                                        |00022600
141   2  2  |GRAPHPL='0'B;                                                          |00022700
142   2  2  |SUBSTR(BUFFER,I,7)='       ';                                         |00022800
143   2  2  |END;                                                                   |00022900

144   2  1  |I=INDEX(BUFFER,'COMPTST');                                             |00022910
145   2  1  |COMPTST='0'B;                                                          |00022920
146   2  1  |IF I>0 THEN DO;                                                        |00022930
147   2  2  |COMPTST='1'B;                                                          |00022940
148   2  2  |SUBSTR(BUFFER,I,7)='       ';                                         |00022950
149   2  2  |END;                                                                   |00022960

150   2  1  |IF INDEX(BUFFER,'=')¬=0 THEN                                           |00022999

           |GET STRING(SUBSTR(BUFFER,7)) DATA(MAXNODES,TI,TL,DEBUG,GOPARM);        |00023000
151   2  1  |END;                                                                   |00023100
           |/* NOW READ IN REST OF CONTROL CARDS FOR THIS NETWORK. */               |00023200


152   2  0  |BEGIN;  /* ALLOCATE AUTOMATIC VARIABLES. */                           |00023300
153   3  0  |DCL 1 NODE_TABLE(MAXNODES),                                            |00023400
           |2 NODE_NUMBER FIXED BIN(31),                                           |00023500
           |2 NODE_POINTER POINTER,                                                |00023600
           |2 DNODE_POINTER POINTER,                                               |00023700
           |[illegible],                                                           |00023800
           |PABORT(3) BIT(1),                                                      |00023900
           |MPTR FIXED BIN(15),                                                    |00024000
           |SPPOS FIXED BIN(15),                                                   |00024100
           |(ITEMP,JTEMP) FIXED BIN(31),                                           |00024200
           |FTEMP FLOAT BIN,                                                       |00024300
           |NODE_MAYBE FIXED BIN(31),                                              |00024400
           |MANHOLE_AREA FLOAT,                                                    |00024500
           |MANHOLE_MAYBE FIXED BIN(31),                                           |00024600
           |JUNCT_MAYBE FIXED BIN(31),                                             |00024700
           |(I,J) FIXED BIN(15),                                                   |00024800
           |ABORT BIT(1),                                                          |00024900
           |(BASE_FLOW, TIME_LAG) FLOAT BIN,                                       |00025000
           |/*                                                                     |00025100
           |(PEAK_FLOW,TIME_TO_PEAK,TIME_TO_CONC) FLOAT BIN,                       |00025200
           |*/                                                                     |00025300
           |GOTO_PLACE LABEL,                                                      |00025400
           |EXIT LABEL EXTERNAL,                                                   |00025410
           |1 DTREE BASED(DTREE_BASE),                                             |00025500
           |2 UP(2) POINTER,                                                       |00025600
           |2 DOWN POINTER,                                                        |00025700
           |ABORT_RUN LABEL INIT(FLUSH);                                           |00025800

154   3  0  |EXIT=EXITC;                                                            |00025890
155   3  0  |ROOT_NODE_#=-1;                                                        |00025900
156   3  0  |#_NODES_ALLOCATED=0;                                                   |00026000


157   3  0  |ON ENDFILE(SYSIN) BEGIN;                                               |00026100
158   4  0  |PUT EDIT('**WARNING:  MISSING "CEND" CARD.  ONE WILL BE GENERATED.',   |00026200
           |'CEND','(GENERATED CONTROL CARD)')                                     |00026300
           |(SKIP,A,SKIP,X(20),A,COL(40),A);                                       |00026400
159   4  0  |DCL CENDSTR CHAR(80) INIT('CEND');    BUFPTR=ADDR(CENDSTR);            |00026450
161   4  0  |GOTO CEND_CARD_RECEIVED;                                               |00026500
162   4  0  |END;                                                                   |00026600
```

```
163   3  0  |GET_CCARD:  READ FILE(SYSIN) SET(BUFPTR);                                                    |00026700
164   3  0  |GOT_CCARD:  PUT FILE(SYSPRINT) EDIT(BUFFER)(SKIP(2),X(20),A);                               |00026800
165   3  0  |REVERT CONVERSION;                                                                          |00026810
                                                                                                         |
166   3  0  |IF SUBSTR(BUFFER,1,6)='START ' THEN DO;                                                     |00026820
167   3  1  |CALL ERRMES('NEXT "START" CARD FOUND BEFORE REQUIRED "CEND" CARD');                         |00026830
168   3  1  |GOTO EXIT;                                                                                  |00026840
169   3  1  |END;                                                                                        |00026850
                                                                                                         |
170   3  0  |SUBSTR(BUFFER,73)=',A';                                                                     |00026900
171   3  0  |IF SUBSTR(BUFFER,1,5)='NODE ' THEN DO;                                                      |00027000
172   3  1  |BUFFER=SUBSTR(BUFFER,6);                                                                    |00027100
173   3  1  |PUT EDIT('*NODES DECLARED:')(SKIP,A);                                                       |00027200
174   3  1  |GET_NEXT_NODE:  STRPOS=VERIFY(BUFFER,', ');                                                 |00027300
175   3  1  |IF SUBSTR(BUFFER,STRPOS,1)='A' THEN GOTO GET_CCARD;                                         |00027400
176   3  1  |BUFFER=SUBSTR(BUFFER,STRPOS);                                                               |00027500
177   3  1  |GET STRING(BUFFER) LIST(NODE_MAYBE);                                                        |00027600
178   3  1  |PUT EDIT(SUBSTR(BUFFER,1,VERIFY(BUFFER,'0123456789')-1))(X(1),A);                           |00027700
           |/* NODE-MAYBE SHOULD NOW HAVE IN IT THE NODE # BEING DECLARED. */                           |00027800
179   3  1  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                                            |00027900
180   3  2  |IF NODE_TABLE.NODE_NUMBER(NTPTR)=NODE_MAYBE THEN DO;                                        |00028000
181   3  3  |CALL ERRMES('LAST NODE LISTED WAS PREVIOUSLY DECLARED');                                    |00028100
182   3  3  |GOTO EXIT;                                                                                  |00028200
183   3  3  |END;                                                                                        |00028300
184   3  2  |END;                                                                                        |00028400
           |/* IF WE GET TO HERE, THE NODE NUMBER IS NOT A DUPLICATE. */                                |00028500
185   3  1  |IF #_NODES_ALLOCATED=MAXNODES THEN DO;                                                      |00028600
186   3  2  |CALL ERRMES('TOO MANY NODES.  ADJUST "MAXNODES" FIGURE ON THE "START" C|00028700
           |ARD');                                                                                      |00028800
187   3  2  |GOTO EXIT;                                                                                  |00028900
188   3  2  |END;                                                                                        |00029000
189   3  1  |ALLOCATE NODE SET(TEMPPTR);                                                                 |00029100
190   3  1  |NODE_PTR=TEMPPTR;                                                                           |00029200
191   3  1  |DO=UPSTREAM_NODEPTR, UPSTREAM_NODEPTR(1), UPSTREAM_NODEPTR(2),                              |00029300
           |YBLOCK_POINTER, FUNCTION_INFO.FUNCPTR,                                                      |00029400
           |FLOWBLOCK_POINTER=NULL;                                                                     |00029500
192   3  1  |IMAGINARY, CRASHED,                                                                         |00029600
           |MANHOLE, Y_JUNCTION, INPUT_STATION, ROOT_STATION ='0'B;                                     |00029700
193   3  1  |FUNCTION_INFO.FUNCVARA, FUNCTION_INFO.FUNCVARB,                                             |00029710
           |TABLE_PAGE#,MANHOLAREA=0;                                                                   |00029800
194   3  1  |#_NODES_ALLOCATED=#_NODES_ALLOCATED+1;                                                      |00029900
195   3  1  |NODE_#,                                                                                     |00030000
           |NODE_TABLE.NODE_NUMBER(#_NODES_ALLOCATED)=NODE_MAYBE;                                       |00030100
196   3  1  |NODE_TABLE.NODE_POINTER(#_NODES_ALLOCATED)=TEMPPTR;                                         |00030200
197   3  1  |ALLOCATE DTREE SET(TEMPPTR);                                                                |00030300
198   3  1  |TEMPPTR->DTREE.UP=NULL;                                                                     |00030400
199   3  1  |TEMPPTR->DTREE.DOWN=NULL;                                                                   |00030500
200   3  1  |NODE_TABLE.DNODE_POINTER(#_NODES_ALLOCATED)=TEMPPTR;                                        |00030600
201   3  1  |GOTO_PLACE=GET_NEXT_NODE;                                                                   |00030700
202   3  1  |GOTO FUNNY_GOSUB;                                                                           |00030800
203   3  1  |END;                                                                                        |00030900
           |/*  NOTE:  EACH NODE MUST BE DECLARED BEFORE THE FIRST USE OF IT ON */                       |00031000
           |/* ANY OTHER TYPE OF CONTROL CARD. */                                                       |00031100
204   3  0  |IF SUBSTR(BUFFER,1,8)='MANHOLE ' THEN DO;                                                   |00031200
205   3  1  |BUFFER=SUBSTR(BUFFER,9);                                                                    |00031300
206   3  1  |BUFFER=TRANSLATE(BUFFER,' ','=');/* CHANGE ANY EQUAL SIGNS TO BLANKS */                    |00031400
207   3  1  |MANHOLE_AREA=10;  /* DEFAULT VALUE FOR MANHOLE-TYPE JUNCTIONS */                            |00031500
208   3  1  |PUT EDIT('*MANHOLES:')(SKIP,A);                                                             |00031600
209   3  1  |GET_NEXT_MANHOLE:  STRPOS=VERIFY(BUFFER,', ');                                              |00031700
210   3  1  |IF SUBSTR(BUFFER,STRPOS,1)='A' THEN IF SUBSTR(BUFFER,STRPOS,4)^='AREA'                      |00031800
           |THEN GOTO GET_CCARD;                                                                        |00031900
211   3  1  |ELSE DO;                                                                                    |00032000
212   3  2  |SUBSTR(BUFFER,STRPOS,4)='    ';                                                            |00032100
213   3  2  |BUFFER=SUBSTR(BUFFER,VERIFY(BUFFER,', '));                                                  |00032200
214   3  2  |GET STRING(BUFFER) LIST(MANHOLE_AREA);                                                      |00032300
215   3  2  |GOTO_PLACE=GET_NEXT_MANHOLE;  GOTO FUNNY_GOSUB;  END;                                       |00032400
216   3  1  |BUFFER=SUBSTR(BUFFER,STRPOS);                                                               |00032500
217   3  1  |GET STRING(BUFFER) LIST(MANHOLE_MAYBE);                                                     |00032600
220   3  1  |PUT EDIT(SUBSTR(BUFFER,1,VERIFY(BUFFER,'0123456789')-1))(X(1),A);                           |00032700
221   3  1  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                                            |00032800
222   3  2  |IF NODE_TABLE.NODE_NUMBER(NTPTR)=MANHOLE_MAYBE THEN DO;                                     |00032900
223   3  3  |TEMPPTR=NODE_TABLE.NODE_POINTER(NTPTR);                                                     |00033000
224   3  3  |TEMPPTR->MANHOLE='1'B;                                                                      |00033100
225   3  3  |TEMPPTR->MANHOLAREA=MANHOLE_AREA;                                                           |00033200
226   3  3  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                                         |00033300
227   3  3  |SCAN_TO_NEXT_MANHOLE:                                                                       |00033400
           |GOTO_PLACE=GET_NEXT_MANHOLE;                                                                |00033500
228   3  3  |GOTO FUNNY_GOSUB;                                                                           |00033600
229   3  3  |END;                                                                                        |00033700
230   3  2  |END;                                                                                        |00033800
231   3  1  |CALL ERRMES('LAST MANHOLE LISTED HAS NOT BEEN DECLARED AS A NODE');                         |00033900
232   3  1  |GOTO EXIT;                                                                                  |00034000
233   3  1  |END;                                                                                        |00034100
```

```
234   3  0  |IF SUBSTR(BUFFER,1,7)='YJUNCT ' THEN DO;                                    |00034200
235   3  1  |BUFFER=SUBSTR(BUFFER,8);                                                     |00034300
236   3  1  |PUT EDIT('**Y-JUNCTIONS:')(SKIP,A);                                          |00034400
237   3  1  |TYPE='Y-JUNCTION';                                                           |00034500
238   3  1  |DCL TYPE CHAR(14) VAR;                                                       |00034600

239   3  1  |GET_NEXT_JUNCT:  STRPOS=VERIFY(BUFFER,', ');                                |00034700
240   3  1  |IF SUBSTR(BUFFER,STRPOS,1)='A' THEN GOTO GET_CCARD;                         |00034800
241   3  1  |BUFFER=SUBSTR(BUFFER,STRPOS);                                               |00034900
242   3  1  |GET STRING(BUFFER) LIST(JUNCT_MAYBE);                                       |00035000
243   3  1  |PUT EDIT(SUBSTR(BUFFER,1,VERIFY(BUFFER,'0123456789')-1))(X(1),A);           |00035100
244   3  1  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                            |00035200
245   3  2  |IF NODE_TABLE.NODE_NUMBER(NTPTR)=JUNCT_MAYBE THEN DO;                       |00035300
246   3  3  |TEMPPTR=NODE_TABLE.NODE_POINTER(NTPTR);                                     |00035400
247   3  3  |IF TYPE¬='Y-JUNCTION' THEN TEMPPTR->IMAGINARY='1'B;  ELSE                   |00035500
248   3  3  |TEMPPTR->Y_JUNCTION='1'B;                                                   |00035600
249   3  3  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                         |00035700
250   3  3  |GOTO PLACE=GET_NEXT_JUNCT;                                                  |00035800
251   3  3  |GOTO FUNNY_GOSUB;                                                           |00035900
252   3  3  |END;                                                                        |00036000
253   3  2  |END;                                                                        |00036100
254   3  1  |CALL ERRMES('LAST '||TYPE||' LISTED HAS NOT BEEN DECLARED A NODE');         |00036200
255   3  1  |GOTO EXIT;                                                                  |00036300

256   3  1  |END;                                                                        |00036400


257   3  0  |IF SUBSTR(BUFFER,1,10)='IMAGINARY ' THEN DO;                                |00036500
258   3  1  |BUFFER=SUBSTR(BUFFER,11);                                                   |00036600
259   3  1  |PUT EDIT('**IMAGINARY NODES:')(SKIP,A);                                     |00036700
260   3  1  |TYPE='IMAGINARY NODE';                                                      |00036800
261   3  1  |GOTO GET_NEXT_JUNCT;                                                        |00036900
262   3  1  |END;                                                                        |00037000
263   3  0  |IF SUBSTR(BUFFER,1,5)='ROOT ' THEN DO;                                      |00037100
264   3  1  |IF ROOT_NODE_#¬=-1 THEN /* ROOT ALREADY DECLARED. */ DO;                    |00037200
265   3  2  |CALL ERRMES('THE ROOT NODE HAS ALREADY BEEN SPECIFIED');                     |00037300
266   3  2  |GOTO EXIT;                                                                  |00037400
267   3  2  |END;                                                                        |00037500
268   3  1  |BUFFER=SUBSTR(BUFFER,6,66);                                                 |00037600
269   3  1  |IF BUFFER=' ' THEN DO;                                                      |00037700
270   3  2  |CALL ERRMES('NO ROOT NODE NUMBER WAS SPECIFIED');                           |00037800
271   3  2  |GOTO EXIT;                                                                  |00037900
272   3  2  |END;                                                                        |00038000
273   3  1  |GET STRING(BUFFER) LIST(ROOT_NODE_#);                                       |00038100
274   3  1  |DO I=1 TO #_NODES_ALLOCATED;                                                |00038200
275   3  2  |IF NODE_NUMBER(I)=ROOT_NODE_# THEN DO;                                      |00038300
276   3  3  |ROOT_POINTER=NODE_POINTER(I);                                               |00038400
277   3  3  |ROOT_POINTER->ROOT_STATION='1'B;                                            |00038500
278   3  3  |IF BADNODE(ROOT_POINTER) THEN GOTO EXIT;                                    |00038600
279   3  3  |GOTO GET_CCARD;                                                             |00038700
280   3  3  |END;                                                                        |00038800
281   3  2  |END;                                                                        |00038900
282   3  1  |CALL ERRMES('SPECIFIED ROOT HAS NOT BEEN PREVIOUSLY DECLARED TO BE A NO|00039000
           |DE');                                                                       |00039100
283   3  1  |GOTO EXIT;                                                                  |00039200
284   3  1  |END;                                                                        |00039300

285   3  0  |IF SUBSTR(BUFFER,1,3)='PD ' THEN DO;                                        |00039400
286   3  1  |ON CONVERSION BEGIN;  /* THIS MEANS AN ERROR OR MISSING VALUE. */          |00039500
287   4  1  |CALL ERRMES('INVALID CHARACTER OR MISSING ENTRY(SHOULD BE 7 OR 8) ON PI|00039600
           |PE DESCRIPTOR CARD');                                                       |00039700
288   4  1  |GOTO EXIT;                                                                  |00039800
289   4  1  |END;                                                                        |00039900
290   3  1  |SUBSTR(BUFFER,73)=',50,A';   /* SETS 50 AS DEFAULT DELTA_X. */              |00040000
291   3  1  |BUFFER=SUBSTR(BUFFER,4);                                                    |00040100
292   3  1  |PUT EDIT('**PIPE GOES FROM NODE #')(SKIP,A);                                |00040200
293   3  1  |BUFFER=SUBSTR(BUFFER,VERIFY(BUFFER,' '));                                   |00040300
294   3  1  |GET STRING(BUFFER) LIST(ITEMP);                                             |00040400
295   3  1  |PUT EDIT(ITEMP)(F(1+(ITEMP>9)+(ITEMP>99)+(ITEMP>999)+(ITEMP>9999)));        |00040500
296   3  1  |DO I=1 TO #_NODES_ALLOCATED;                                                |00040600
297   3  2  |IF NODE_TABLE.NODE_NUMBER(I)=ITEMP THEN GOTO READ_TONODE;                   |00040700
298   3  2  |END;                                                                        |00040800
299   3  1  |UNDECLARED_NODE:  CALL ERRMES('NODE SPECIFIED HAS NOT BEEN DECLARED');      |00040900
300   3  1  |GOTO EXIT;                                                                  |00041000
301   3  1  |READ_TONODE:  GOTO_PLACE=OK_TO_READ_TONODE;                                 |00041100
302   3  1  |GOTO FUNNY_GOSUB;                                                           |00041200
303   3  1  |OK_TO_READ_TONODE:                                                          |00041300
           |BUFFER=SUBSTR(BUFFER,VERIFY(BUFFER,', '));                                  |00041400
304   3  1  |GET STRING(BUFFER) LIST(JTEMP);                                             |00041500
305   3  1  |PUT EDIT(' TO NODE #')(A);                                                  |00041590
306   3  1  |PUT EDIT(JTEMP)(F(1+(JTEMP>9)+(JTEMP>99)+(JTEMP>999)+(JTEMP>9999)));        |00041600
307   3  1  |IF ITEMP=JTEMP THEN DO;                                                     |00041700
308   3  2  |CALL ERRMES('PIPES ARE NOT ALLOWED TO RETURN TO THEIR UPSTREAM NODE');      |00041800
309   3  2  |GOTO EXIT;                                                                  |00041900
310   3  2  |END;                                                                        |00042000
311   3  1  |DO J=1 TO #_NODES_ALLOCATED ;                                               |00042100
312   3  2  |IF NODE_TABLE.NODE_NUMBER(J)=JTEMP THEN GOTO TONODE_IS_OK;                  |00042200
313   3  2  |END;                                                                        |00042300
314   3  1  |GOTO UNDECLARED_NODE;                                                       |00042400
```

```
315   3  1  |TONODE_IS_OK;                                                                  |00042500
            |TEMPPTR=NODE_TABLE.NODE_POINTER(J);   /*POINTS TO DOWNSTREAM NODE */          |00042600
316   3  1  |IF TEMPPTR->NODE.UPSTREAM_NODEPTR(1)¬=NULL THEN IF                            |00042700
            |TEMPPTR->NODE.UPSTREAM_NODEPTR(2)=NULL THEN DO;                               |00042800
317   3  2  |ITEMP=2;                                                                      |00042900
318   3  2  |TEMPPTR->NODE.UPSTREAM_NODEPTR(2)=NODE_TABLE.NODE_POINTER(I);                 |00043000
319   3  2  |END;                                                                          |00043100
320   3  1  |ELSE DO;                                                                      |00043200
321   3  2  |CALL ERRMES('TWO PIPES ALREADY ENTER THE SPECIFIED JUNCTION');                |00043300
322   3  2  |GOTO EXIT;                                                                    |00043400
323   3  2  |END;                                                                          |00043500
324   3  1  |ELSE DO;                                                                      |00043600
325   3  2  |TEMPPTR->NODE.UPSTREAM_NODEPTR(1)=NODE_TABLE.NODE_POINTER(I);                 |00043700
326   3  2  |ITEMP=1;                                                                      |00043800
327   3  2  |END;                                                                          |00043900
328   3  1  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                           |00044000
329   3  1  |TEMPPTR=NODE_TABLE.DNODE_POINTER(J);  /* POINTS TO DOWNSTREAM DTREE */        |00044100
330   3  1  |TEMPPTR->DTREE.UP(ITEMP)=NODE_TABLE.DNODE_POINTER(I);                         |00044200
331   3  1  |TEMPPTR=NODE_TABLE.NODE_POINTER(I);                                           |00044300
            |/* POINTS TO UPSTREAM NODE. */                                              |00044400
332   3  1  |IF TEMPPTR->NODE.DOWNSTREAM_NODEPTR¬=NULL THEN DO;                            |00044500
333   3  2  |CALL ERRMES('A PIPE ALREADY LEAVES THE SPECIFIED UPSTREAM STATION');          |00044600
334   3  2  |GOTO EXIT;                                                                    |00044700
335   3  2  |END;                                                                          |00044800
336   3  1  |TEMPPTR->NODE.DOWNSTREAM_NODEPTR=NODE_TABLE.NODE_POINTER(J);                  |00044900
337   3  1  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                           |00045000
338   3  1  |TEMPPTR=NODE_TABLE.DNODE_POINTER(I);  /* POINTS TO UPSTREAM DTREE */         |00045100
339   3  1  |TEMPPTR->DTREE.DOWN=NODE_TABLE.DNODE_POINTER(J);                              |00045200
            |/* BOTH NODE TREE AND DTREE TREES ARE (OR SHOULD BE) PROPERLY CHAINED*/      |00045300
340   3  1  |TEMPPTR=NODE_TABLE.NODE_POINTER(I);  /* POINTS AGAIN TO UPSTREAM NODE*/       |00045400
341   3  1  |GOTO_PLACE=OK_TO_READ_PIPEDATA;                                               |00045500
342   3  1  |GOTO FUNNY_GOSUB;                                                             |00045600
343   3  1  |OK_TO_READ_PIPEDATA:                                                          |00045700
            |BUFFER=SUBSTR(BUFFER,VERIFY(BUFFER,', '));                                    |00045800
344   3  1  |GET STRING(BUFFER) LIST(TEMPPTR->NODE.PIPE_LENGTH, TEMPPTR->NODE.             |00045900
            |PIPE_SLOPE,TEMPPTR->NODE.PIPE_DIAMETER,TEMPPTR->NODE.PIPE_ROUGHNESS,          |00046000
            |TEMPPTR->NODE.PIPE_DROP,TEMPPTR->DX);                                         |00046100
345   3  1  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                           |00046200
            |/* IF WE GET HERE, PRELIMINARY INSPECTION OF THE NODE LOOKS LIKE IT'S*/       |00046300
            |/* OK, AT LEAST SO FAR. */                                                   |00046400
346   3  1  |REVERT CONVERSION;                                                            |00046500
347   3  1  |GOTO GET_CCARD;                                                               |00046600
348   3  1  |END;                                                                          |00046700
349   3  0  |IF SUBSTR(BUFFER,1,4)='FBD ' THEN DO;                                        |00051300
350   3  1  |I, NTPTR=0;                                                                   |00051400
351   3  1  |GOTO CHOP_FBD;  END;                                                          |00051500
353   3  0  |IF SUBSTR(BUFFER,1,5)='JFBD ' THEN DO;                                       |00051600
354   3  1  |I, NTPTR=1;                                                                   |00051700
355   3  1  |CHOP_FBD:  NTPTR=NTPTR+5;                                                     |00051800
356   3  1  |BUFFER=SUBSTR(BUFFER,NTPTR);                                                  |00051900
357   3  1  |ON CONVERSION BEGIN;  /* THIS MEANS AN ERROR OR MISSING VALUE. */            |00052000
358   4  1  |CALL ERRMES('INVALID CHARACTER OR MISSING ENTRY(SHOULD BE 4) ON FLOWBLO       |00052100
            |CK DESCRIPTOR CARD');                                                         |00052200
359   4  1  |GOTO EXIT;                                                                    |00052300
360   4  1  |END;                                                                          |00052400
361   3  1  |PUT EDIT('*FLOWBLOCK DESCRIPTION FOR NODE #')(SKIP,A);                        |00052500
362   3  1  |STRPOS=VERIFY(BUFFER,', ');                                                   |00052600
363   3  1  |IF SUBSTR(BUFFER,STRPOS,1)='A' THEN MISSING_FBD_VALUES:  SIGNAL CONVERS       |00052700
            |ION;                                                                          |00052800
364   3  1  |BUFFER=SUBSTR(BUFFER,STRPOS);                                                 |00052900
365   3  1  |GET STRING(BUFFER) LIST(NODE_MAYBE);                                          |00053000
366   3  1  |PUT EDIT(SUBSTR(BUFFER,1,VERIFY(BUFFER,'0123456789')-1))(X(1),A);             |00053100
367   3  1  |PUT EDIT('.  (TIME LAG, DURATION, BASE FLOW.)')(A);                           |00053200
368   3  1  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                              |00053300
369   3  2  |IF NODE_TABLE.NODE_NUMBER(NTPTR)=NODE_MAYBE THEN GOTO GOT_FBD_NODE;           |00053400
370   3  2  |END;                                                                          |00053500
371   3  1  |CALL ERRMES('SPECIFIED NODE HAS NOT BEEN PREVIOUSLY DECLARED');               |00053600
372   3  1  |GOTO EXIT;                                                                    |00053700
373   3  1  |GOT_FBD_NODE:  TEMPPTR=NODE_TABLE.NODE_POINTER(NTPTR);                        |00053800
374   3  1  |IF I=0 THEN DO;                                                               |00053900
375   3  2  |TEMPPTR->INPUT_STATION='1'B;                                                  |00054000
376   3  2  |IF BADNODE(TEMPPTR) THEN GOTO EXIT;                                           |00054100
377   3  2  |END;                                                                          |00054200
378   3  1  |IF TEMPPTR->FLOWBLOCK_POINTER¬=NULL THEN DO;                                  |00054300
379   3  2  |CALL ERRMES('INPUT FOR THIS NODE HAVE ALREADY BEEN DECLARED');                |00054400
380   3  2  |GOTO EXIT;                                                                    |00054500
381   3  2  |END;                                                                          |00054600
382   3  1  |GOTO_PLACE=GET_FILEBLOCK_D_PARMS;                                             |00054700
383   3  1  |GOTO FUNNY_GOSUB;                                                             |00054800
384   3  1  |GET_FILEBLOCK_D_PARMS:  GET STRING(BUFFER) LIST(TIME_LAG,DURATION,BASE_       |00054900
            |FLOW);                                                                        |00055000
            |/* NEXT ALLOCATE THE FLOWBLOCK. */                                          |00055100
385   3  1  |ALLOCATE FLOWBLOCK(TIME_LAG/TI:(TIME_LAG+DURATION)/TI);                       |00055200
386   3  1  |IF I=0 THEN                                                                   |00055300
            |LASTALLOC,TEMPPTR->FLOWBLOCK_POINTER=PSEUDOREGISTER;                          |00055400
387   3  1  |ELSE LASTALLOC, TEMPPTR->YBLOCK_POINTER=PSEUDOREGISTER;                       |00055500
            |/* NEXT:  INITIALIZE THE ARRAY. */                                          |00055600
            |/* NOTE THAT AT THIS POINT, FLOWBLOCK==YBLOCK. */                           |00055700
388   3  1  |FLOWBLOCK=BASE_FLOW;                                                          |00055800
            |/* PREPARE TO READ VALUES INTO FLOWBLOCK. */                                |00055900
389   3  1  |ON CONVERSION BEGIN;                                                          |00056000
```

```
390   4  1  |DCL STR CHAR(150) VAR;  STR=ONSOURCE;                                                    |00056100
392   4  1  |IF INDEX(STR,'FEND')>0 | INDEX(STR,'JEND')>0 THEN GOTO GET_CCARD;                        |00056200
            |/* READY TO GET NEXT CCARD. */                                                          |00056700
393   4  1  |CALL ERRMES('"'||ONSOURCE||'" CANNOT BE CHANGED TO A NUMBER.  THE "FEND|00056800
            |" OR "JEND" CARD MAY BE MISSING');                                                      |00056900
394   4  1  |GOTO EXIT;                                                                              |00057100
395   4  1  |END;                                                                                    |00057200


396   3  1  |READ FILE(SYSIN) SET(BUFPTR);                                                           |00057300
397   3  1  |J=0;  /* WHEN J=0, WE'RE READING TIME VALUE, WHEN J=1, FLOW VALUE. */                  |00057305
398   3  1  |DO WHILE(VERIFY(SUBSTR(BUFFER,1,1),'ABCDEFGHIJKLMNOPQRSTUVWXYZ')=1);                    |00057310
399   3  2  |PUT EDIT(BUFFER)(SKIP,X(20),A);                                                         |00057315
400   3  2  |SUBSTR(BUFFER,73,8)=',A';                                                               |00057320
401   3  2  |GETFBDVAL:  STRPOS=VERIFY(BUFFER,', ');                                                 |00057325
402   3  2  |IF SUBSTR(BUFFER,STRPOS,1)¬='A' THEN DO;                                                |00057330
403   3  3  |BUFFER=SUBSTR(BUFFER,STRPOS);                                                           |00057335
404   3  3  |IF J=0 THEN DO;  /* READ IN A TIME VALUE. */                                          |00057340
405   3  4  |GET STRING(BUFFER) LIST(TIME_VALUE);                                                    |00057345
406   3  4  |TIME_VALUE=TIME_VALUE/TI;  /* GET ACTUAL FLOWBLOCK SUBSCRIPT. */                       |00057350
407   3  4  |J=1;  /* SO NEXT READ WILL GET THE FLOWBLOCK VALUE. */                                  |00057355
408   3  4  |END;                                                                                    |00057360
409   3  3  |ELSE DO;  /* READ IN A FLOW VALUE. */                                                 |00057365
410   3  4  |IF TIME_VALUE<LBOUND(FLOWBLOCK,1) | TIME_VALUE>HBOUND(FLOWBLOCK,1)                     |00057370
            |THEN                                                                                    |00057375
            |PUT EDIT('* WARNING:  TIME SPECIFIED,',TIME_VALUE*TI,', IS OUTSIDE THE |00057380
            |RANGE SPECIFIED ON THE CONTROL CARD.','THE PAIR OF VALUES WILL NOT BE U|00057385
            |SED.')                                                                                  |00057390
            |(SKIP,A,P'ZZZ,ZZ9',A,SKIP,X(12),A);                                                     |00057395
411   3  4  |ELSE GET STRING(BUFFER) LIST(FLOWBLOCK(TIME_VALUE));                                    |00057400
412   3  4  |J=0; /* SO NEXT READ WILL READ A NEW TIME VALUE. */                                     |00057405
413   3  4  |END;                                                                                    |00057410
414   3  3  |GOTO_PLACE=GETFBDVAL;                                                                   |00057415
415   3  3  |GOTO FUNNY_GOSUB;                                                                       |00057420
416   3  3  |END;                                                                                    |00057425
417   3  2  |READ FILE(SYSIN) SET(BUFPTR);                                                           |00057430
418   3  2  |END;                                                                                    |00057435
419   3  1  |IF SUBSTR(BUFFER,1,5)='JEND ' | SUBSTR(BUFFER,1,5)='FEND ' THEN DO;                    |00057440
420   3  2  |PUT EDIT(BUFFER)(SKIP,X(20),A);                                                         |00057445
421   3  2  |GOTO GET_CCARD;                                                                         |00057450
422   3  2  |END;                                                                                    |00057455
423   3  1  |ELSE GOTO GOT_CCARD;                                                                    |00057460
424   3  1  |END /* OF THE "FBD" COMMAND CARD HANDLER. */ ;                                          |00058200

425   3  0  |IF SUBSTR(BUFFER,1,15)='OUTLET CONTROL ' THEN BEGIN;                                    |00058205
426   4  0  |DCL FCTN_STR CHAR(250) VAR,                                                             |00058210
            |1 SYMTB(4),                                                                             |00058215
            |2 NAME CHAR(12) INIT('TIME', 'DISCHARGE','VELOCITY','DEPTH'),                           |00058220
            |2 PTR POINTER;                                                                          |00058225

427   4  0  |FCTN_STR=SUBSTR(BUFFER,16,57);                                                          |00058230
428   4  0  |GET_CONT_CARD:  READ FILE(SYSIN) SET(BUFPTR);                                           |00058235
429   4  0  |DO WHILE(SUBSTR(BUFFER,1,1)=' ');                                                       |00058240
430   4  1  |PUT FILE(SYSPRINT) EDIT(BUFFER)(SKIP,X(20),A);                                          |00058245
431   4  1  |FCTN_STR=FCTN_STR||SUBSTR(BUFFER,2,71);                                                 |00058250
432   4  1  |READ FILE(SYSIN) SET(BUFPTR);                                                           |00058255
433   4  1  |END;                                                                                    |00058260

434   4  0  |SYMTB.PTR(1)=ADDR(OUTVARS.FT);                                                          |00058265
435   4  0  |SYMTB.PTR(2)=ADDR(OUTVARS.FDI);                                                         |00058270
436   4  0  |SYMTB.PTR(3)=ADDR(OUTVARS.FV);                                                          |00058275
437   4  0  |SYMTB.PTR(4)=ADDR(OUTVARS.FDE);                                                         |00058280
438   4  0  |IF OUTVARS.FUNCPTR=NULL THEN DO;                                                        |00058281
439   4  1  |FCTN_STR=SUBSTR(FCTN_STR,VERIFY(FCTN_STR,' '));                                         |00058282
440   4  1  |OUTDPTH=0;  /* GIVE IT AN INITIAL VALUE. */                                            |00058283
441   4  1  |I=VERIFY(FCTN_STR,'0123456789. ');                                                      |00058284
442   4  1  |IF I¬=1 THEN DO;                                                                        |00058285
443   4  2  |OUTDPTH=SUBSTR(FCTN_STR,1,I-1);                                                         |00058286
444   4  2  |SUBSTR(FCTN_STR,1,I-1)=' ';                                                             |00058287
445   4  2  |END;                                                                                    |00058288
446   4  1  |DCL OUTDPTH FLOAT BIN EXTERNAL;                                                         |00058289
447   4  1  |OUTVARS.FUNCPTR=COMPILE(FCTN_STR,OUTVARS.FUNCVARA,OUTVARS.FUNCVARR,                     |00058290
            |SYMTB);                                                                                 |00058295
448   4  1  |END;                                                                                    |00058297
449   4  0  |ELSE DO;                                                                                |00058300
450   4  1  |CALL ERRMES('THE OUTLET CONTROL FUNCTION HAS ALREADY BEEN SPECIFIED');                   |00058305
451   4  1  |GOTO EXIT;  END;                                                                        |00058310

453   4  0  |GOTO GOT_CCARD;                                                                         |00058315

454   4  0  |END;                                                                                    |00058320
```

```
455   3  0  |IF SUBSTR(BUFFER,1,5)='CEND ' THEN DO;                                          |00058325
456   3  1  |CEND_CARD_RECEIVED:                                                              |00058400
            |ABORT='0'B;                                                                     |00058500
457   3  1  |IF ROOT_NODE_#=-1 THEN DO;                                                       |00058600
458   3  2  |CALL ERRMES('NO ROOT NODE HAS BEEN SPECIFIED');                                  |00058700
459   3  2  |GOTO EXIT;                                                                       |00058800
460   3  2  |END;                                                                            |00058900
            |/* VERIFY EACH NODE OF THE TREE, THEN THE TREE ITSELF, THEN RETURN */           |00059000
461   3  1  |PUT EDIT('*NODE VALIDITY CHECKING BEGINS.')(SKIP,A);                             |00059100
462   3  1  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                                 |00059200
463   3  2  |TPTR1=NODE_TABLE.NODE_POINTER(NTPTR);                                           |00059300
464   3  2  |IF BADNODE(TPTR1) THEN ABORT='1'B;                                              |00059400
465   3  2  |I=(TPTR1->MANHOLE='1'B)+(TPTR1->Y_JUNCTION='1'B)                                |00059500
            |+(TPTR1->IMAGINARY='1'B)                                                        |00059600
            |+(TPTR1->INPUT_STATION='1'B)+(TPTR1->ROOT_STATION='1'B);                        |00059700
466   3  2  |IF I=0 THEN DO;                                                                 |00059800
467   3  3  |CALL ERRMES('YOU DID NOT SPECIFY OF WHICH TYPE NODE #'||TPTR1->NODE_#           |00059900
            |||' IS');                                                                       |00060000
468   3  3  |ABORT='1'B;                                                                     |00060100
469   3  3  |END;                                                                            |00060200
470   3  2  |IF TPTR1->INPUT_STATION THEN IF TPTR1->FLOWBLOCK_POINTER=NULL THEN DO;          |00060300
471   3  3  |CALL ERRMES('NO FLOWBLOCK DESCRIPTOR CARD WAS FOUND FOR THE INPUT STATI|00060400
            |ON AT NODE'||TPTR1->NODE_#);                                                    |00060500
472   3  3  |ABORT='1'B;                                                                     |00060600
473   3  3  |END;                                                                            |00060700
474   3  2  |IF TPTR1->IMAGINARY THEN DO;                                                    |00060800
475   3  3  |IF TPTR1->FLOWBLOCK_POINTER=NULL THEN DO;                                       |00060900
476   3  4  |PUT EDIT('*WARNING* NO FLOW WAS SPECIFIED INTO IMAGINARY NODE, #',              |00061000
            |TPTR1->NODE_#,'.')(SKIP,A,P'ZZ,ZZ9',A(1));                                     |00061100
            |/* NEXT CREATE A DUMMY FLOWBLOCK SO SHOWTREE WON'T BE CONFUSED. */            |00061200
477   3  4  |ALLOCATE FLOWBLOCK(-2:-1);                                                      |00061300
478   3  4  |LASTALLOC, TPTR1->FLOWBLOCK_POINTER=PSEUDOREGISTER;                             |00061400
479   3  4  |FLOWBLOCK=0;                                                                    |00061500
480   3  4  |END;                                                                            |00061600
481   3  3  |TPTR1->DX, TPTR1->PIPE_LENGTH, TPTR1->PIPE_ROUGHNESS,                           |00061700
            |TPTR1->PIPE_DIAMETER, TPTR1->PIPE_SLOPE=1;                                      |00061800
482   3  3  |TPTR1->PIPE_DROP=1000;                                                          |00061900
483   3  3  |IF TPTR1->YBLOCK_POINTER¬=NULL THEN DO;                                         |00062000
484   3  4  |CALL ERRMES('A JPBD CARD WAS ILLEGALLY GIVEN FOR IMAGINARY NODE, #'             |00062100
            |||TPTR1->NODE_#);                                                              |00062200
485   3  4  |ABORT='1'B;                                                                     |00062300
486   3  4  |END;                                                                            |00062400
487   3  3  |END;                                                                            |00062500
488   3  2  |IF TPTR1->DOWNSTREAM_NODEPTR=NULL THEN IF TPTR1->ROOT_STATION='0'B              |00062600
            |THEN DO;                                                                       |00062700
489   3  3  |CALL ERRMES('NODE'||TPTR1->NODE_#||' HAS NO OUTGOING PIPE');                    |00062800
490   3  3  |ABORT='1'B;                                                                     |00062900
491   3  3  |END;                                                                            |00063000
492   3  2  |ELSE DO;  /* IT IS A ROOT NODE. */                                             |00063100
493   3  3  |TPTR1->FUNCTION_INFO=OUTVARS, BY NAME;                                          |00063110
494   3  3  |IF TPTR1->UPSTREAM_NODEPTR(1)=NULL THEN DO;                                     |00063120
495   3  4  |CALL ERRMES('ROOT NODE HAS NO INCOMING PIPE');                                  |00063200
496   3  4  |ABORT='1'B;                                                                     |00063300
497   3  4  |END;                                                                            |00063400
498   3  3  |END;                                                                            |00063410
499   3  2  |IF TPTR1->MANHOLE THEN GOTO NODE_IS_A_JUNCTION;                                 |00063500
500   3  2  |IF TPTR1->Y_JUNCTION THEN NODE_IS_A_JUNCTION:  DO;                              |00063600
501   3  3  |PABORT='0'B;                                                                    |00063700
502   3  3  |IF TPTR1->UPSTREAM_NODEPTR(1)=NULL THEN PABORT(1)='1'B;                         |00063800
503   3  3  |IF TPTR1->UPSTREAM_NODEPTR(2)=NULL THEN PABORT(2)='1'B;                         |00063900
504   3  3  |IF TPTR1->DOWNSTREAM_NODEPTR=NULL THEN PABORT(3)='1'B;                          |00064000
505   3  3  |IF PABORT(1)|PABORT(2)|PABORT(3) THEN DO;                                       |00064100
506   3  4  |MSGBUF=' ';                                                                     |00064200
507   3  4  |PUT STRING(MSGBUF) EDIT('JUNCTION AT NODE #',TPTR1->NODE_#,                     |00064300
            |' IS LACKING PIPE(S): ',' UPSTREAM(1)',' UPSTREAM(2)',                         |00064400
            |' DOWNSTREAM')                                                                 |00064500
            |(A,P'ZZ,ZZ9',A,A((PABORT(1)='1'B)*12),A((PABORT(2)='1'B)*12),                   |00064600
            |A((PABORT(3)='1'B)*11),A(1));                                                  |00064700
508   3  4  |CALL ERRMES(MSGBUF);                                                            |00064800
509   3  4  |ABORT='1'B;                                                                     |00064900
510   3  4  |END;                                                                            |00065000
511   3  3  |END;                                                                            |00065100
512   3  2  |IF TPTR1->IMAGINARY THEN DO;                                                    |00065200
513   3  3  |TPTR2=TPTR1->DOWNSTREAM_NODEPTR;                                                |00065300
514   3  3  |IF TPTR2¬=NULL THEN DO;                                                         |00065400
515   3  4  |IF TPTR2->UPSTREAM_NODEPTR(2)¬=TPTR1 THEN DO;                                   |00065500
516   3  5  |TPTR3=TPTR2->UPSTREAM_NODEPTR(2);                                               |00065600
517   3  5  |TPTR2->UPSTREAM_NODEPTR(2)=TPTR2->UPSTREAM_NODEPTR(1);                          |00065700
518   3  5  |TPTR2->UPSTREAM_NODEPTR(1)=TPTR3;                                               |00065800
519   3  5  |IF TPTR3¬=NULL THEN DO;                                                         |00065900
520   3  6  |IF TPTR3->IMAGINARY THEN DO;                                                    |00066000
521   3  7  |CALL ERRMES('BOTH UPSTREAM NODES AT NODE #'||TPTR2->NODE_#||' ARE IMAGI|00066100
            |NARY');                                                                       |00066200
522   3  7  |ABORT='1'B;                                                                     |00066300
523   3  7  |END;                                                                            |00066400
524   3  6  |END;                                                                            |00066500
525   3  5  |END;                                                                            |00066600
526   3  4  |END;                                                                            |00066700
527   3  3  |END;                                                                            |00066800
528   3  2  |END;                                                                            |00066900
529   3  1  |IF ABORT THEN GOTO EXIT;                                                        |00067000
```

```
530  3  1  |PUT EDIT('*NODE VALIDITY CHECKING HAS COMPLETED SUCCESSFULLY.  TREE VAL|00067100
          |IDITY CHECKING BEGINS.') (SKIP,A);                                     |00067200
531  3  1  |ABORT='1'B;                                                            |00067300
532  3  1  |PUT EDIT('*DELETING NODES:') (SKIP,A);                                 |00067400
533  3  1  |#_NODES_LEFT=#_NODES_ALLOCATED;                                        |00067500
534  3  1  |DO WHILE(ABORT);                                                       |00067600
          |/* NOTE THAT HERE ABORT CHANGES ROLES.  HEREAFTER, ABORT IS USED AS A*/|00067700
          |/* DELETION FLAG. */                                                  |00067800
535  3  2  |ABORT='0'B;                                                            |00067900
536  3  2  |DO NTPTR=1 TO #_NODES_ALLOCATED;                                       |00068000
537  3  3  |TPTR1=NODE_TABLE.DNODE_POINTER(NTPTR);                                 |00068100
538  3  3  |IF TPTR1¬=NULL THEN DO;                                                |00068200
539  3  4  |I=(TPTR1->UP(1)=NULL)+(TPTR1->UP(2)=NULL)+(TPTR1->DOWN=NULL);          |00068300
540  3  4  |IF I=2 THEN DO;                                                       |00068400
          |/* IT IS AN END-OF-BRANCH NODE. */                                    |00068500
541  3  5  |IF TPTR1->UP(1)¬=NULL THEN DO;                                         |00068600
542  3  6  |TPTR2=TPTR1->UP(1);                                                    |00068700
543  3  6  |TPTR2->DOWN=NULL;                                                      |00068800
544  3  6  |END;                                                                   |00068900
545  3  5  |ELSE IF TPTR1->UP(2)¬=NULL THEN DO;                                    |00069000
546  3  6  |TPTR2=TPTR1->UP(2);                                                    |00069100
547  3  6  |TPTR2->DOWN=NULL;                                                      |00069200
548  3  6  |END;                                                                   |00069300
549  3  5  |ELSE IF TPTR1->DOWN¬=NULL THEN DO;                                     |00069400
550  3  6  |TPTR2=TPTR1->DOWN;                                                     |00069500
551  3  6  |IF TPTR2->UP(1)=TPTR1 THEN TPTR2->UP(1)=NULL;                          |00069600
552  3  6  |ELSE IF TPTR2->UP(2)=TPTR1 THEN TPTR2->UP(2)=NULL;                     |00069700
553  3  6  |ELSE SIGNAL ERROR;                                                     |00069800
554  3  6  |END;                                                                   |00069900
          |/* THE NODE TO BE DELETED IS NOW LOGICALLY DE-CHAINED. */              |00070000
555  3  5  |FREE TPTR1->DTREE;                                                     |00070100
556  3  5  |NODE_TABLE.DNODE_POINTER(NTPTR)=NULL;                                  |00070200
557  3  5  |#_NODES_LEFT=#_NODES_LEFT-1;                                           |00070300
558  3  5  |ABORT='1'B;                                                            |00070400
559  3  5  |TPTR1=NODE_TABLE.NODE_POINTER(NTPTR);                                  |00070500
560  3  5  |PUT EDIT(TPTR1->NODE_#,' ')(P(6),A(1));                                |00070600
561  3  5  |END;                                                                   |00070700
562  3  4  |END;                                                                   |00070800
563  3  3  |END;                                                                   |00070900
564  3  2  |END;                                                                   |00071000
          |/* WHEN WE GET HERE, ONLY ONE DTREE NODE SHOULD BE LEFT, AND ALL */    |00071100
          |/* THREE POINTERS IN IT SHOULD BE NULL. */                            |00071200
565  3  1  |IF #_NODES_LEFT=1 THEN DO;                                             |00071300
566  3  2  |DO I=1 TO #_NODES_ALLOCATED;                                           |00071400
567  3  3  |IF NODE_TABLE.DNODE_POINTER(I)¬=NULL THEN DO;                          |00071500
568  3  4  |TPTR1=NODE_TABLE.DNODE_POINTER(I);                                     |00071600
569  3  4  |IF TPTR1->UP(1)¬=NULL | TPTR1->UP(2)¬=NULL | TPTR1->DOWN ¬= NULL THEN,|00071700
          |SIGNAL ERROR;                                                         |00071800
570  3  4  |FREE TPTR1->DTREE;                                                     |00071900
571  3  4  |NODE_TABLE.DNODE_POINTER(I)=NULL;                                      |00072000
572  3  4  |#_NODES_LEFT=#_NODES_LEFT-1;                                           |00072100
573  3  4  |GOTO TREE_CHECKS_OK;                                                   |00072200
574  3  4  |END;                                                                   |00072300
575  3  3  |END;                                                                   |00072400
576  3  2  |END;                                                                   |00072500
577  3  1  |ELSE DO;                                                               |00072600
          |/* TOO MUCH TREE IS LEFT. */                                          |00072700
578  3  2  |IF #_NODES_LEFT=0 THEN SIGNAL ERROR;                                   |00072800
579  3  2  |PUT EDIT('*****ERROR.  TREE HAS ONE OR MORE CLOSED LOOPS.  THE ASSOCIAT|00072900
          |ED NODES ARE:') (SKIP,A);                                              |00073000
580  3  2  |DO I=1 TO #_NODES_ALLOCATED;                                           |00073100
581  3  3  |TPTR1=NODE_TABLE.NODE_POINTER(I);                                      |00073200
582  3  3  |IF NODE_TABLE.DNODE_POINTER(I)¬=NULL THEN                              |00073300
          |PUT EDIT(TPTR1->NODE_#)(SKIP,COL(5),P(6));                             |00073400
583  3  3  |END;                                                                   |00073500
584  3  2  |PUT EDIT('*EVALUATION OF THIS NETWORK IS BEING ABORTED.')(SKIP(2),A);  |00073600
585  3  2  |ABORT_RUN=START_READING_CONTROL_CARDS;                                 |00073700
586  3  2  |GOTO EXIT;                                                             |00073800
587  3  2  |END;                                                                   |00073900
588  3  1  |TREE_CHECKS_OK:  PUT EDIT('*TREE VALIDITY CHECKING HAS COMPLETED SUCCES|00074000
          |SFULLY.  READY FOR FLOW EVALUATION.') (SKIP,A);                        |00074100
589  3  1  |CALL MAKE_TABLE(ROOT_POINTER);                                         |00074200
590  3  1  |IF EXECUTE THEN;  ELSE DO;                                             |00074300
592  3  2  |PUT EDIT('*EXECUTION IS BEING BYPASSED AS REQUESTED.')(SKIP(2),A);     |00074400
593  3  2  |ABORT_RUN=START_READING_CONTROL_CARDS;                                 |00074500
594  3  2  |GOTO EXIT;                                                             |00074600
595  3  2  |END;                                                                   |00074700
596  3  1  |ALLOCATE NODE_LIST;                                                    |00074800
597  3  1  |DO I=1 TO #_NODES_ALLOCATED;                                           |00074900
598  3  2  |NODE_LIST(I)=NODE_TABLE.NODE_POINTER(I);                               |00075000
599  3  2  |END;                                                                   |00075100
600  3  1  |RETURN;                                                                |00075200
601  3  1  |END;                                                                   |00075300
602  3  0  |IF SUBSTR(BUFFER,1,1)=' ' THEN DO;                                     |00075400
603  3  1  |CALL ERRMES('CONTROL CARD TYPE NOT FOUND STARTING IN COLUMN 1');       |00075500
604  3  1  |GOTO EXIT;                                                             |00075600
605  3  1  |END;                                                                   |00075700
606  3  0  |CALL ERRMES('UNRECOGNIZED CONTROL CARD TYPE');                         |00075800
607  3  0  |GOTO EXIT;                                                             |00075900
```

```
                |/* THIS NEXT BIT OF CODE FREES ALL ALLOCATED STORAGE, BEFORE FLUSHING*/|00076000
608    3   0     |EXITC:  DO I=1 TO #_NODES_ALLOCATED;                                  |00076100
609    3   1     |TEMPPTR=NODE_TABLE.NODE_POINTER(I);                                   |00076200
610    3   1     |FREE TEMPPTR->NODE;                                                   |00076300
611    3   1     |TEMPPTR=NODE_TABLE.DNODE_POINTER(I);                                  |00076400
612    3   1     |IF TEMPPTR¬=NULL THEN FREE TEMPPTR->DTREE;                            |00076500
613    3   1     |END;                                                                  |00076600
614    3   0     |DO WHILE (ALLOCATION(FLOWBLOCK));                                     |00076700
615    3   1     |FREE FLOWBLOCK;                                                       |00076800
616    3   1     |END;                                                                  |00076900
617    3   0     |LASTALLOC=PSEUDOREGISTER;                                             |00077000
618    3   0     |REVERT CONVERSION;                                                    |00077100
619    3   0     |GOTO ABORT_RUN;                                                       |00077300


                |/* NEXT PART IS A PSEUDO-SUBROUTINE. */                              |00077400
620    3   0     |DCL (M,N) FIXED BIN(15);                                              |00077500
621    3   0     |FUNNY_GOSUB:                                                          |00077600
                |M=INDEX(BUFFER,',');                                                  |00077700
622    3   0     |N=INDEX(BUFFER,' ');                                                  |00077800
623    3   0     |IF N=0 THEN N=4092;                                                   |00077900
624    3   0     |IF M=0 THEN M=4092;                                                   |00077910
625    3   0     |IF M=N /* ONLY OCCURS IF BOTH ARE 4092 */ THEN SIGNAL ERROR;         |00078000
626    3   0     |BUFFER=SUBSTR(BUFFER,MIN(M,N));                                       |00078200
627    3   0     |GOTO GOTO_PLACE;                                                      |00078300

628    3   0     |END;  /* OF BEGIN BLOCK. */                                          |00078400
629    2   0     |END SETUP;                                                            |00078500

630    1   0     |MAKE_TABLE:  PROC(PTR_TO_ROOT);                                       |00078600
631    2   0     |DCL PTR_TO_ROOT POINTER,                                              |00078700
                |;                                                                     |00078900
632    2   0     |SIGNAL ENDPAGE(SYSPRINT);                                             |00079000
633    2   0     |PUT EDIT('PIPES SUMMARY TABLE')(COL(56),A);                           |00079100
634    2   0     |CALL PIPE_HEADINGS;                                                   |00079200
635    2   0     |CALL PIPEPRINT(PTR_TO_ROOT);                                          |00079300


636    2   0     |PIPEPRINT:  PROC(RPTR) RECURSIVE;                                     |00079400
637    3   0     |DCL I FIXED BIN(31),                                                  |00079500
                |(TEMPPTR,RPTR,N1PTR,N2PTR) POINTER,                                   |00079600
                |;                                                                     |00079700
638    3   0     |N1PTR,N2PTR=NULL;                                                     |00079800
639    3   0     |DO I=1 TO 2;                                                          |00079900
640    3   1     |TEMPPTR=RPTR->UPSTREAM_NODEPTR(I);                                    |00080000
                |/* TEMPPTR NOW POINTS TO ONE UPSTREAM NODE. */                        |00080100
641    3   1     |IF TEMPPTR¬=NULL THEN IF ¬TEMPPTR->INPUT_STATION &¬TEMPPTR->IMAGINARY |00080200
                |/* I.E. THERE A) IS THIS UPSTREAM NODE, AND B) NOT AN INPUT STA. */   |00080300
                |THEN DO;                                                              |00080400
642    3   2     |CALL PIPEPRINT(TEMPPTR);                                              |00080500
643    3   2     |IF I=1 THEN N1PTR=TEMPPTR;                                            |00080600
644    3   2     |ELSE N2PTR=TEMPPTR;                                                   |00080700
645    3   2     |END;                                                                  |00080800
646    3   1     |ELSE /* NODE EXISTS, HAS NO UPSTREAM STATIONS.  */                    |00080900
                |IF I=1 THEN N1PTR=TEMPPTR;                                            |00081000
647    3   1     |ELSE N2PTR=TEMPPTR;                                                   |00081100
648    3   1     |ELSE DO;                                                              |00081200
                |/* RPTR POINTS TO A REAL ROOT NODE. */                               |00081300
649    3   2     |TEMPPTR=RPTR->UPSTREAM_NODEPTR(1);                                    |00081400
650    3   2     |CALL PDATAPRINT(TEMPPTR,RPTR);                                        |00081500
651    3   2     |RETURN;                                                               |00081600
652    3   2     |END;                                                                  |00081700
653    3   1     |END;                                                                  |00081800
654    3   0     |CALL PDATAPRINT(N1PTR,RPTR);                                          |00081900
655    3   0     |CALL PDATAPRINT(N2PTR,RPTR);                                          |00082000
656    3   0     |END PIPEPRINT;                                                        |00082100


657    2   0     |PDATAPRINT:  PROC(UPTR,DPTR);                                         |00082200
658    3   0     |DCL (UPTR,DPTR) POINTER;                                              |00082300
659    3   0     |IF LINENO(SYSPRINT)>50 THEN DO;                                       |00082400
660    3   1     |SIGNAL ENDPAGE(SYSPRINT);                                             |00082500
661    3   1     |CALL PIPE_HEADINGS;                                                   |00082600
662    3   1     |END;                                                                  |00082700
663    3   0     |PUT EDIT(UPTR->NODE_#, DPTR->NODE_#, UPTR->PIPE_LENGTH,               |00082800
                |UPTR->PIPE_SLOPE, UPTR->PIPE_DIAMETER, UPTR->PIPE_ROUGHNESS,          |00082900
                |UPTR->PIPE_DROP,UPTR->DX)                                             |00083000
                |(SKIP(2), COL(24), P'ZZ,ZZ9', COL(35), P'ZZ,ZZ9', COL(45),           |00083100
                |P'ZZZ,ZZ9', COL(57), P'V.999999', COL(68), P'ZZ9V.99', COL(80),       |00083200
                |P'V.9999', COL(92), P'9V.9', COL(102), P'ZZZ9');                      |00083300
664    3   0     |END PDATAPRINT;                                                       |00083400
```

```
665   2  0  |PIPE_HEADINGS:  PROC;                                                                    |00083500
666   3  0  |PUT EDIT('PIPE GOES', 'LENGTH', 'DIAMETER', 'DROP', 'DELTA_X',                           |00083600
           |'FROM NODE', 'TO NODE', 'IN FEET', 'SLOPE', 'IN FEET', 'ROUGHNESS',                      |00083700
           |'IN FEET',                                                                               |00083800
           |'IN FEET','_________','_________','_________','_________','_________',                  |00083900
           |'_________','_________','_________')                                                    |00084000
           |(SKIP(2), COL(29), A, COL(46), A, COL(68), A, COL(91), A, COL(101),                      |00084100
           |A, COL(23), A, COL(35), A, COL(46), A, COL(58), A, COL(68), A,                           |00084200
           |COL(78), A, COL(90), A, COL(101), A, SKIP(0), COL(23), B(A,X(2)));                       |00084300
667   1  0  |END PIPE_HEADINGS;                                                                       |00084400

668   2  0  |END MAKE_TABLE;                                                                          |00084500




669   1  0  |LASTPIPE:  PROC(A);                                                                      |00085300
670   2  0  |DCL A POINTER;                                                                           |00085400
671   2  0  |PUT EDIT('*SUBROUTINE "LASTPIPE" HAS BEEN CALLED.  THE PASSED POINTER P                  |00085500
           |OINTS TO NODE #',A->NODE_#)(SKIP,A,F(6));                                                |00085600
           |/* THIS IS A TEMPORARY SUBROUTINE TO BE REPLACED LATER BY THE REAL ONE.                  |00085700
           |*/                                                                                       |00085800
672   2  0  |TPTR1=A->FLOWBLOCK_POINTER;                                                              |00085900
673   2  0  |CALL PLUCK(TPTR1,LASTALLOC);                                                             |00086000
674   2  0  |PSEUDOREGISTER=TPTR1;                                                                    |00086100
675   2  0  |FREE FLOWBLOCK;                                                                          |00086200
676   2  0  |LASTALLOC=PSEUDOREGISTER;                                                                |00086300
677   2  0  |END LASTPIPE;                                                                            |00086400


678   1  0  |END_MAIN:  END NETWORK;                                                                  |00086500
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	#_NODES_ALLOCATED	AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0) 1,156,179,185,194,194,195,196,200,221,244,274,296,311,368,462,533,536,566, 580,596,597,608
2	#_NODES_LEFT	AUTOMATIC ALIGNED BINARY FIXED (15,0) 533,557,557,565,572,572,578
570	A	/* PARAMETER */ ALIGNED POINTER 671,672
153	ABORT	AUTOMATIC UNALIGNED BIT (1) 456,464,468,472,485,490,496,509,522,529,531,534,535,558
153	ABORT_RUN	AUTOMATIC ALIGNED INITIAL LABEL 152,585,593,619
********	ADDR	BUILTIN 5,160,434,435,436,437
********	ALLOCATION	BUILTIN 24,614
51	BADNODE	ENTRY RETURNS(BIT (1)) 226,249,278,328,337,345,376,464
153	BASE_FLOW	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 384,388
94	BUFFER	BASED (BUFPTR) UNALIGNED CHARACTER (80) 106,116,117,118,120,121,126,127,130,132,136,138,142,144,148,150,150 101,102,102,164,166,170,171,172,172,174,175,176,176,177,178,178,204,205,205, 206,206,209,210,210,212,213,213,213,214,218,218,219,220,220,234,235,235,239, 240,241,241,242,243,243,257,258,258,263,268,268,269,273,285,290,291,291,293, 293,293,294,303,303,303,304,343,343,343,344,349,353,356,356,362,363,364,364, 365,366,366,384,398,399,400,401,402,403,403,405,411,419,419,420,425,455,602, 621,622,626,626 427,429,430,431
94	BUFPTR	AUTOMATIC ALIGNED POINTER 105,106,116,117,118,120,121,122,126,127,130,132,136,138,142,144,148,150,150 101,101,102,102,163,164,166,170,171,172,172,174,175,176,176,177,178,178,204, 205,205,206,206,209,210,210,212,213,213,213,214,218,218,219,220,220,234,235, 235,239,240,241,241,242,243,243,257,258,258,263,268,268,269,273,285,290,291, 291,293,293,293,294,303,303,303,304,343,343,343,344,349,353,356,356,362,363, 364,364,365,366,366,384,396,398,399,400,401,402,403,403,405,411,417,419,419, 420,425,455,602,621,622,626,626 160,427,428,429,430,431,432
13	BUILD_THE_TREE	/* STATEMENT LABEL CONSTANT */ 27
455	CEND_CARD_RECEIVED	/* STATEMENT LABEL CONSTANT */ 161
159	CENDSTR	AUTOMATIC UNALIGNED INITIAL CHARACTER (80) 157,160
355	CHOP_FBD	/* STATEMENT LABEL CONSTANT */ 351
94	COMPILE	EXTERNAL ENTRY(UNALIGNED CHARACTER(*),ALIGNED BINARY FIXED (15,0),ALIGNED BINARY FIXED (15,0),*) RETURNS(POINTER) 447
94	COMPTST	STATIC EXTERNAL UNALIGNED BIT (1) 145,147
2	CRASHED	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1) 192
2	CRITICAL_DEPTH	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DATE	BUILTIN 3
94	DCOMPIL	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 112

```
2          DEBUG                    STATIC EXTERNAL UNALIGNED CHARACTER (20) VARYING
                                    13,150

153        DNODE_POINTER            (*) /* IN NODE_TABLE */ AUTOMATIC ALIGNED POINTER
                                    200,329,330,338,339,537,556,567,568,571,582,611

2          DNORM                    /* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)

153        DOWN                     /* IN DTREE */ BASED ALIGNED POINTER
                                    199,339,539,543,547,549,550,569

2          DOWNSTREAM_NODEPTR       /* IN NODE */ BASED ALIGNED POINTER
                                    70,191,332,336,488,504,513

2          DOWNSTREAM_PIPE_INFO     /* IN NODE */ BASED /* STRUCTURE */

658        DPTR                     /* PARAMETER */ ALIGNED POINTER
                                    663

153        DTREE                    BASED (DTREE_BASE) /* STRUCTURE */
                                    197,555,570,612

********   DTREE_BASE               AUTOMATIC ALIGNED POINTER

********   DURATION                 AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                    384,385

2          DX                       /* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    344,481,663

678        END_MAIN                 /* STATEMENT LABEL CONSTANT */
                                    98

2          ERRMES                   EXTERNAL ENTRY(UNALIGNED CHARACTER(*)) RETURNS(DECIMAL /* SINGLE */ FLOAT
                                    (6))
                                    62,66,71,75,80,84,88,119
                                    102,167,181,186,231,254,265,270,282,299,308,321,333,371,379,458,467,471,484,
                                    489,495,508,521,603,606
                                    287,358,393,450

2          EVALUBR                  EXTERNAL ENTRY(ALIGNED POINTER ,ALIGNED POINTER ,ALIGNED POINTER )
                                     RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                    49

94         EXECUTE                  AUTOMATIC UNALIGNED BIT (1)
                                    113,129,590

153        EXIT                     STATIC EXTERNAL ALIGNED LABEL
                                    154,168,182,187,226,232,249,255,266,271,278,283,300,309,322,328,334,337,345,
                                    372,376,380,459,529,586,594,604,607
                                    288,359,394,451

508        EXITC                    /* STATEMENT LABEL CONSTANT */
                                    154

426        FCTN_STR                 AUTOMATIC UNALIGNED CHARACTER (250) VARYING
                                    427,431,431,439,439,439,441,443,444,447

94         FDE                      /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    437

94         FDI                      /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    435

2          FLOWBLOCK                (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    24,25,385,385,388,410,410,411,477,479,614,615,675

2          FLOWBLOCK_POINTER        /* IN NODE */ BASED ALIGNED POINTER
                                    33,74,191,378,386,470,475,478,672

120        FLUSH                    /* STATEMENT LABEL CONSTANT */
                                    103,152

94         FT                       /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    434

153        FTEMP                    AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

94         FUNCPTR                  /* IN OUTVARS EXTERNAL */ STATIC ALIGNED POINTER
                                    110,493
                                    438,447

2          FUNCPTR                  /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                    POINTER
                                    191,493
```

```
2          FUNCTION_INFO              /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */

2          FUNCVARA                   /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                      BINARY FIXED (15,0)
                                      193,493

94         FUNCVARA                   /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY FIXED (15,0)
                                      111,493
                                      447

2          FUNCVARB                   /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                      BINARY FIXED (15,0)
                                      193,493

94         FUNCVARB                   /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY FIXED (15,0)
                                      111,493
                                      447

621        FUNNY_GOSUB                /* STATEMENT LABEL CONSTANT */
                                      202,216,228,251,302,342,383,415

94         FV                         /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      436

163        GET_CCARD                  /* STATEMENT LABEL CONSTANT */
                                      175,210,240,279,347,421
                                      392

428        GET_CONT_CARD              /* STATEMENT LABEL CONSTANT */

384        GET_FILEBLOCK_D_PARMS      /* STATEMENT LABEL CONSTANT */
                                      382

239        GET_NEXT_JUNCT             /* STATEMENT LABEL CONSTANT */
                                      250,261

209        GET_NEXT_MANHOLE           /* STATEMENT LABEL CONSTANT */
                                      215,227

174        GET_NEXT_NODE              /* STATEMENT LABEL CONSTANT */
                                      201

401        GETFBDVAL                  /* STATEMENT LABEL CONSTANT */
                                      414

2          GETFBPR                    EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                      5

2          GETTFPR                    EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          GETTGPR                    EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          GETTBPR                    EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          GOPARM                     STATIC EXTERNAL UNALIGNED CHARACTER (30) VARYING
                                      14,150

164        GOT_CCARD                  /* STATEMENT LABEL CONSTANT */
                                      423,453

373        GOT_FBD_NODE               /* STATEMENT LABEL CONSTANT */
                                      369

106        GOT_FIRST_CONTROL_CARD     /* STATEMENT LABEL CONSTANT */
                                      121

153        GOTO_PLACE                 AUTOMATIC ALIGNED LABEL
                                      201,215,227,250,301,341,382,414,627

2          GRAPHPL                    STATIC EXTERNAL UNALIGNED BIT (1)
                                      139,141

********   HBOUND                     BUILTIN
                                      19,410

52         HOLDING                    AUTOMATIC UNALIGNED CHARACTER (95) VARYING
                                      55,56,56,57,57,58,58,59,59,60,60,61,61,61,62

153        I                          AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                      274,274,275,276,296,296,297,313,325,330,331,338,340,350,354,374,386,465,466,
                                      539,540,566,566,567,568,571,580,580,581,582,597,597,598,598,608,608,609,611
                                      441,442,443,444

29         I                          AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                      31,31,32,35,38,42

637        I                          AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                      639,639,640,643,646
```

```
52          I                       AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                    51,53,54

********    I                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                    19,19,20,127,128,130,132,134,136,138,140,142,144,146,148

2           IMAGINARY               /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                    53,59,83,87,192,247,465,474,512,520,641

********    INDEX                   BUILTIN
                                    127,132,138,144,150
                                    102,621,622
                                    392,392

2           INPUT_STATION           /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                    53,58,79,87,192,375,465,470,641

153         ITEMP                   AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                    294,295,295,295,295,295,297,307,317,326,330

153         J                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                    311,311,312,315,329,336,339,397,404,407,412

29          JNODE_PTR               AUTOMATIC ALIGNED POINTER
                                    30,48,49

153         JTEMP                   AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                    304,306,306,306,306,306,307,312

153         JUNCT_MAYBE             AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                    242,245

2           LASTALLOC               STATIC EXTERNAL ALIGNED POINTER
                                    15,386,387,478,617,673,676

569         LASTPIPE                ENTRY RETURNS(BINARY FIXED (15,0))
                                    44
********    LBOUND                  BUILTIN
                                    410

********    LENGTH                  BUILTIN
                                    61

********    LINENO                  BUILTIN
                                    659

620         M                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                    621,624,624,625,626

630         MAKE_TABLE              ENTRY RETURNS(BINARY FIXED (15,0))
                                    18,589

2           MANHOLAREA              /* IN NODE */ BASED ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                    193,225

2           MANHOLE                 /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                    53,56,192,224,465,499

153         MANHOLE_AREA            AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                    207,214,225

153         MANHOLE_MAYBE           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                    219,222

2           MAXNODES                AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                    1,107,150,152,185

********    MIN                     BUILTIN
                                    626

363         MISSING_FBD_VALUES      /* STATEMENT LABEL CONSTANT */

94          MSGBUF                  AUTOMATIC UNALIGNED CHARACTER (100) VARYING
                                    506,507,508

620         N                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                    622,623,623,625,626

426         NAME                    (4) /* IN SYMTB */ AUTOMATIC UNALIGNED INITIAL CHARACTER (12)
                                    425,425,425,425

2           NET#                    AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                    1,115,115,116
                                    97

1           NETWORK                 EXTERNAL ENTRY RETURNS(BINARY FIXED (15,0))
```

```
2          NODE                    BASED (NODE_PTR) /* STRUCTURE */
                                   21,189,610

2          NODE_#                  /* IN NODE */ BASED ALIGNED BINARY FIXED (16,0)
                                   195,467,471,476,484,489,507,521,560,582,663,663,671

503        NODE_IS_A_JUNCTION      /* STATEMENT LABEL CONSTANT */
                                   499

2          NODE_LIST               (*) CONTROLLED EXTERNAL ALIGNED POINTER
                                   19,20,23,596,596,598

153        NODE_MAYBE              AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                   177,180,195,195,365,369

153        NODE_NUMBER             (*) /* IN NODE_TABLE */ AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                   180,195,222,245,275,297,312,369

153        NODE_POINTER            (*) /* IN NODE_TABLE */ AUTOMATIC ALIGNED POINTER
                                   196,223,246,276,315,318,325,331,336,340,373,463,559,581,598,609

********   NODE_PTR                AUTOMATIC ALIGNED POINTER
                                   190,191,191,191,191,191,191,192,192,192,192,192,192,193,193,193,193,195

153        NODE_TABLE              (*)AUTOMATIC /* STRUCTURE */

2          NODE_TYPE               /* IN NODE */ BASED /* STRUCTURE */

52         NODEPTR                 /* PARAMETER */ ALIGNED POINTER
                                   53,53,53,53,53,56,57,58,59,60,65,65,70,74,79,79,83,83,87,87,87,87

29         NODE1_PTR               AUTOMATIC ALIGNED POINTER
                                   30,35,38,49

29         NODE2_PTR               AUTOMATIC ALIGNED POINTER
                                   30,36,39,49

153        NTPTR                   AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                   179,179,180,221,221,222,223,244,244,245,246,350,354,355,355,356,368,368,369,
                                   373,462,462,463,536,536,537,556,559

2          NULL                    BUILTIN
                                   15,65,70,74,79,83,87,110,191,198,199,315,316,332,378,470,475,483,488,494,
                                   502,503,504,514,519,538,539,539,539,541,543,545,547,549,551,552,556,567,569,
                                   569,569,571,582,612
                                   438,638,641

29         NULL                    BUILTIN
                                   30,33,33

637        N1PTR                   AUTOMATIC ALIGNED POINTER
                                   638,643,646,654

637        N2PTR                   AUTOMATIC ALIGNED POINTER
                                   638,644,647,655

343        OK_TO_READ_PIPEDATA     /* STATEMENT LABEL CONSTANT */
                                   341

303        OK_TO_READ_TONODE       /* STATEMENT LABEL CONSTANT */
                                   301

2          ONSOURCE                BUILTIN
                                   391,393

446        OUTDPTH                 STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   440,443

94         OUTVARS                 STATIC EXTERNAL /* STRUCTURE */

153        PABORT                  (3) AUTOMATIC UNALIGNED BIT (1)
                                   501,502,503,504,505,505,505,507,507,507

2          PAGEBIT                 AUTOMATIC UNALIGNED INITIAL BIT (1)
                                   1,7,7

2          PAGENO                  STATIC EXTERNAL ALIGNED INITIAL DECIMAL FIXED (4,0)
                                   9,10,10

657        PDATAPRINT              ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                   650,654,655

2          PIPE_DIAMETER           /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                   FLOAT (21)
                                   344,481,663

2          PIPE_DROP               /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                   FLOAT (21)
                                   344,482,663
```

```
665         PIPE_HEADINGS               ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                        634,661

2           PIPE_LENGTH                 /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)
                                        344,481,663

2           PIPE_ROUGHNESS              /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                        FLOAT (21)
                                        344,481,663

2           PIPE_SLOPE                  /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                        FLOAT (21)
                                        344,481,663

535         PIPEPRINT                   ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                        635,642

2           PLUCK                       EXTERNAL ENTRY(ALIGNED POINTER ,ALIGNED POINTER ) RETURNS(DECIMAL
                                        /* SINGLE */ FLOAT (6))
                                        673

********    PSEUDOBASE                  AUTOMATIC ALIGNED POINTER
                                        5,386,387,478,617,674,676

2           PSEUDOREGISTER              BASED (PSEUDOBASE) ALIGNED POINTER
                                        386,387,478,617,674,676

426         PTR                         (4) /* IN SYMTB */ AUTOMATIC ALIGNED POINTER
                                        434,435,436,437

531         PTR_TO_ROOT                 /* PARAMETER */ ALIGNED POINTER
                                        635

301         READ_TONODE                 /* STATEMENT LABEL CONSTANT */
                                        297

2           ROOT_NODE_#                 AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        155,264,273,275,457

29          ROOT_POINTER                /* PARAMETER */ ALIGNED POINTER
                                        32,40,43,48

2           ROOT_POINTER                AUTOMATIC ALIGNED POINTER
                                        17,18,276,277,278,589

2           ROOT_STATION                /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                        40,53,60,65,87,192,277,465,488

537         RPTR                        /* PARAMETER */ ALIGNED POINTER
                                        640,649,650,654,655

227         SCAN_TO_NEXT_MANHOLE        /* STATEMENT LABEL CONSTANT */

2           SETDIAM                     STATIC EXTERNAL UNALIGNED BIT (1)
                                        133,135

93          SETUP                       ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                        16

28          SRCHTREE                    ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                        17,34

105         START_READING_CONTROL_CARDS /* STATEMENT LABEL CONSTANT */
                                        106,124,585,593

390         STR                         AUTOMATIC UNALIGNED CHARACTER (150) VARYING
                                        391,392,392

153         STRPOS                      AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        174,175,176,209,210,210,212,218,239,240,241,362,363,364,401,402,403

********    SUBSTR                      BUILTIN
                                        4,4,4,61,106,117,118,120,121,126,130,136,142,148,150
                                        101,102,166,170,171,172,175,176,178,204,205,210,210,212,213,218,220,234,235,
                                        240,241,243,257,258,263,268,285,290,291,293,303,343,349,353,356,363,364,366,
                                        398,400,402,403,419,419,425,455,602,626
                                        427,429,431,439,443,444

426         SYMTB                       (*)AUTOMATIC /* STRUCTURE */
                                        447

34          SYSIN                       EXTERNAL FILE RECORD SEQUENTIAL INPUT ENVIRONMENT(RECSIZE(80) CONSECUTIVE
                                        TOTAL)
                                        95,105,122
                                        101,157,163,396,417,428,432
```

```
********  SYSPRINT                EXTERNAL FILE PRINT
                                  6,8,9,11,114,116
                                  96,97,164,173,178,208,220,236,243,259,292,295,305,306,361,366,367,393,410,
                                  420,461,476,530,532,560,579,582,584,588,592
                                  158,430,632,633,659,660,663,666,671

2         TABLE_PAGE#           /* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0)
                                  193

153       TEMPPTR               AUTOMATIC ALIGNED POINTER
                                  189,190,196,197,198,199,200,223,224,225,226,246,247,248,249,315,316,316,318,
                                  325,328,329,330,331,332,336,337,338,339,340,344,344,344,344,344,344,345,373,
                                  375,376,378,386,387,609,610,611,612,612

637       TEMPPTR               AUTOMATIC ALIGNED POINTER
                                  640,641,641,641,642,643,644,646,647,649,650

29        TEMPPTR               AUTOMATIC ALIGNED POINTER
                                  32,33,33,34,35,36,38,39,43,44

94        TI                    STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                  109,150,385,385,406,410

153       TIME_LAG              AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                  384,385,385

********  TIME_VALUE            AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                  405,406,406,410,410,410,411

94        TL                    STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                  108,150

2         TODAY                 AUTOMATIC UNALIGNED CHARACTER (8)
                                  3,4,4,4,4,9

315       TONODE_IS_OK          /* STATEMENT LABEL CONSTANT */
                                  312

2         TPTR1                 AUTOMATIC ALIGNED POINTER
                                  20,21,463,464,465,465,465,465,465,467,470,470,471,474,475,476,478,481,481,
                                  481,481,481,482,483,484,488,488,489,493,493,493,494,499,500,502,503,504,507,
                                  512,513,515,537,538,539,539,539,541,542,545,546,549,550,551,552,555,559,560,
                                  568,569,569,569,570,581,582,672,673,674

2         TPTR2                 AUTOMATIC ALIGNED POINTER
                                  513,514,515,516,517,517,518,521,542,543,546,547,550,551,551,552,552

2         TPTR3                 AUTOMATIC ALIGNED POINTER
                                  516,518,519,520

********  TRANSLATE             BUILTIN
                                  206

588       TREE_CHECKS_OK        /* STATEMENT LABEL CONSTANT */
                                  573

238       TYPE                  AUTOMATIC UNALIGNED CHARACTER (14) VARYING
                                  237,247,254,260

299       UNDECLARED_NODE       /* STATEMENT LABEL CONSTANT */
                                  314

153       UP                    (2) /* IN DTREE */ BASED ALIGNED POINTER
                                  198,330,539,539,541,542,545,546,551,551,552,552,569,569

2         UPSTREAM_NODEPTR      (2) /* IN NODE */ BASED ALIGNED POINTER
                                  32,43,65,79,83,191,191,316,316,318,325,494,502,503,515,516,517,517,518,640,
                                  649

558       UPTR                  /* PARAMETER */ ALIGNED POINTER
                                  663,663,663,663,663,663,663

********  VERIFY                BUILTIN
                                  174,178,209,213,220,239,243,293,303,343,362,366,398,401,439,441

2         Y_JUNCTION            /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                  53,57,192,248,465,500

2         YBLOCK_POINTER        /* IN NODE */ BASED ALIGNED POINTER
                                  87,191,387,483
```

```
COMPILE:  PROC(FUNCTION,AVARNO,RVARNO,SYMTB) RETURNS(POINTER) REORDER;
```

SOURCE LISTING

```
STMT LEV NT

    1      0  |COMPILE:  PROC(FUNCTION,AVARNO,RVARNO,SYMTB) RETURNS(POINTER) REORDER; |00000100
    2    1 0  |DCL FUNCTION CHAR(*),                                                  |00000200
              |(AVARNO,RVARNO) FIXED BIN(15),                                         |00000300
              |THREAD_BASE POINTER STATIC INIT(NULL),                                 |00000400
              |P POINTER STATIC,                                                      |00000500
              |PSET POINTER,                                                          |00000600
              |APTR POINTER BASED,                                                    |00000700
              |SIZE INIT(50) FIXED BIN(15),                                           |00000800
              |VALUE POINTER,                                                         |00000900
              |FCTN CHAR(250) VAR,                                                    |00001000
              |PDEPTH FIXED BIN(15),                                                  |00001100
              |C1 CHAR(1),                                                            |00001200
              |NULL BUILTIN,                                                          |00001300
              |ERRMES ENTRY(CHAR(*)) EXTERNAL,                                        |00001400
              |EXIT LABEL EXTERNAL,                                                   |00001500
              |COMPTST BIT(1) EXTERNAL,                                               |00001600
              |DCOMPIL ENTRY EXTERNAL,                                                |00001700
              |VALUATE ENTRY EXTERNAL,                                                |00001800
              |(I,J) FIXED BIN(15),                                                   |00001900
              |1 STATEMENT BASED(P),                                                  |00002000
              |2 THREAD POINTER,                                                      |00002100
              |2 NUM_STEPS FIXED BIN(15),                                             |00002200
              |2 #ELTS FIXED BIN(15),                                                 |00002300
              |2 CODE(SIZE REFER(#ELTS)),                                             |00002400
              |3 OPERATION FIXED BIN(15),                                             |00002500
              |3 OPERAND_ONE POINTER,                                                 |00002600
              |3 OPERAND_TWO POINTER,                                                 |00002700
              |3 RESULT POINTER,                                                      |00002800
              |1 CONSTANTS STATIC,                                                    |00002900
              |2 NUM_CONSTS FIXED BIN(15) INIT(1), /* FIRST ONE IS A CONSTANT 0. */  |00003000
              |2 CONSTANT(50) FLOAT BIN INIT(0),                                      |00003100
              |1 TEMPORARIES STATIC, /* CANNOT BE AUTO SINCE, ALTHOUGH THE VALUES     |00003200
              |NEED NOT BE SAVED FROM ONE EXECUTION TO THE NEXT, STATEMENT STRUCTURE   |00003300
              |CONTAINS ABSOLUTE POINTERS TO THE TEMPORARIES, THEREFORE THEIR LOCA-    |00003400
              |TION MUST NEVER CHANGE. */                                             |00003500
              |2 NUM_TEMPS FIXED BIN(15),                                             |00003600
              |2 TEMPRARY (50) FLOAT BIN,                                             |00003700
              |1 SYMTB(*),                                                            |00003800
              |2 NAME CHAR(12),                                                       |00003900
              |2 PTR POINTER;                                                         |00004000

    3    1 0  |IF THREAD_BASE=NULL THEN DO;                                           |00004100
    4    1 1  |ALLOCATE STATEMENT SET(THREAD_BASE);                                   |00004200
    5    1 1  |PSET=ADDR(THREAD_BASE);                                                |00004300
    6    1 1  |P=THREAD_BASE;                                                         |00004400
    7    1 1  |END;                                                                   |00004500
    8    1 0  |ELSE DO;                                                               |00004600
    9    1 1  |ALLOCATE STATEMENT SET(P->STATEMENT.THREAD);                           |00004700
   10    1 1  |PSET=ADDR(P->STATEMENT.THREAD);                                        |00004800
   11    1 1  |P=P->STATEMENT.THREAD;                                                 |00004900
   12    1 1  |END;                                                                   |00005000
              |/* THIS NEEDS TO BE CHAINED IN RIGHT NOW, SO THAT IF AN ERROR CAUSES   |00005100
              |RETURN THROUGH EXIT, THIS NEW ALLOCATION CAN BE FREED PROPERLY. */     |00005200

   13    1 0  |OPERAND_ONE,OPERAND_TWO,THREAD,RESULT=NULL;                            |00005300
   14    1 0  |PDEPTH,NUM_TEMPS,NUM_STEPS,RVARNO=0;                                   |00005400

   15    1 0  |FCTN=SUBSTR(FUNCTION,VERIFY(FUNCTION,' ')); /* REMOVE LEADING BLANKS*/ |00005500

   16    1 0  |DO I=LENGTH(FCTN) TO 1 BY -1 WHILE(SUBSTR(FCTN,I,1)=' '); END;         |00005600
   17    1 0  |IF I^=LENGTH(FCTN) THEN FCTN=SUBSTR(FCTN,1,I); /* REMOVE TRAILING    |00005700
              |BLANKS. */                                                             |00005800

   18    1 0  |I=INDEX(FCTN,' '); /* FIND ANY CONTAINED BLANKS. */                    |00005900
   20    1 0  |DO J=1 TO 150 WHILE(I>0);                                              |00006000
   21    1 1  |FCTN=SUBSTR(FCTN,1,I-1)||SUBSTR(FCTN,I+VERIFY(SUBSTR(FCTN,I+1),' '));  |00006100
   22    1 1  |I=INDEX(FCTN,' '); END; /* ALL CONTAINED BLANKS ARE NOW GONE. */       |00006200
```

```
24   1  0  |EXPR:  PROC(FCTN) RETURNS(POINTER) RECURSIVE;                          |00006300
25   2  0  |DCL FCTN CHAR(250) VAR,                                                |00006400
           |(PTR1,PTR2) POINTER,                                                   |00006500
           |I FIXED BIN(15),                                                       |00006600
           |C1 CHAR(1),                                                            |00006700
           |PDEPTH FIXED BIN(15) INIT(0);                                          |00006800

26   2  0  |DO I=1 TO LENGTH(FCTN);                                                |00006900

27   2  1  |C1=SUBSTR(FCTN,I,1);                                                   |00007000

28   2  1  |PDEPTH=PDEPTH+(C1='(')-(C1=')');                                       |00007100
29   2  1  |IF PDEPTH<0 THEN DO;                                                   |00007200
30   2  2  |CALL EMSG('EXPRESSION','MANY');                                        |00007300
31   2  2  |GOTO EXIT;                                                             |00007400
32   2  2  |END;                                                                   |00007500

33   2  1  |IF PDEPTH=0 THEN IF C1='+' | C1='-' THEN DO;                          |00007600
34   2  2  |PTR1=TERM(SUBSTR(FCTN,1,I-1));                                         |00007700
35   2  2  |PTR2=EXPR(SUBSTR(FCTN,I+1));                                           |00007800
36   2  2  |RETURN(CODEGEN(PTR1,PTR2,1+(C1='-')));                                 |00007900
37   2  2  |END;                                                                   |00008000

38   2  1  |END;  /* OF LOOP DO I=1 TO LENGTH(FCTN) */                            |00008100

39   2  0  |IF PDEPTH¬=0 THEN DO;  /* MAKE SURE ALL PARENS MATCH UP. */            |00008200
40   2  1  |CALL EMSG('EXPRESSION','FEW');                                         |00008300
41   2  1  |GOTO EXIT;                                                             |00008400
42   2  1  |END;                                                                   |00008500

43   2  0  |RETURN(TERM(FCTN));                                                    |00008600

44   2  0  |END EXPR;                                                              |00008700



45   1  0  |EMSG:  PROC(TYPE,TOO);                                                 |00008800
46   2  0  |DCL (TYPE,TOO) CHAR(10) VAR;                                           |00008900
47   2  0  |CALL ERRMES('BAD PARENTHESIS NESTING IN '||TYPE||', OR TOO '||TOO      |00009000
           |||' RIGHT PARENTHESES');                                               |00009100
48   2  0  |END EMSG;                                                              |00009200



49   1  0  |CODEGEN:  PROC(PTR1,PTR2,OPCODE) RETURNS(POINTER);                     |00009300
50   2  0  |DCL(PTR1,PTR2) POINTER,                                                |00009400
           |OPCODE FIXED BIN(15);                                                  |00009500

51   2  0  |NUM_STEPS=NUM_STEPS+1;                                                 |00009600
52   2  0  |IF NUM_STEPS>HBOUND(CODE.OPERATION,1)-1 THEN TOOBIG:  DO;              |00009700
53   2  1  |CALL ERRMES('EVALUATION REQUIRES TOO MANY STEPS');                     |00009800
54   2  1  |GOTO EXIT;                                                             |00009900
55   2  1  |END;                                                                   |00010000

56   2  0  |CODE(NUM_STEPS).OPERATION=OPCODE;                                      |00010100
57   2  0  |CODE(NUM_STEPS).OPERAND_ONE=PTR1;                                      |00010200
58   2  0  |CODE(NUM_STEPS).OPERAND_TWO=PTR2;                                      |00010300

59   2  0  |NUM_TEMPS=NUM_TEMPS+1;                                                 |00010400
60   2  0  |IF NUM_TEMPS>HBOUND(TEMPRARY,1) THEN GOTO TOOBIG;                      |00010500

61   2  0  |CODE(NUM_STEPS).RESULT=ADDR(TEMPRARY(NUM_TEMPS));                      |00010600
62   2  0  |RETURN(ADDR(TEMPRARY(NUM_TEMPS)));                                     |00010700
63   2  0  |END CODEGEN;                                                           |00010800



64   1  0  |TERM:  PROC(FCTN) RETURNS(POINTER) RECURSIVE;                          |00010900
65   2  0  |DCL FCTN CHAR(250) VAR,                                                |00011000
           |(PTR1,PTR2) POINTER,                                                   |00011100
           |I FIXED BIN(15),                                                       |00011200
           |C1 CHAR(1),                                                            |00011300
           |PDEPTH FIXED BIN(15) INIT(0),                                          |00011400
           |FUNCS(5:10) CHAR(4) STATIC INIT('SIN ','SIND','COS ','COSD','SQRT');   |00011500

66   2  0  |IF FCTN=' ' THEN RETURN(ADDR(CONSTANT(1))); /* RETURN ADDR OF A 0. */  |00011600

67   2  0  |DO I=1 TO LENGTH(FCTN);                                                |00011700

68   2  1  |C1=SUBSTR(FCTN,I,1);                                                   |00011800

69   2  1  |PDEPTH=PDEPTH+(C1='(')-(C1=')');                                       |00011900
70   2  1  |IF PDEPTH<0 THEN DO;                                                   |00012000
71   2  2  |CALL EMSG('TERM','MANY');                                              |00012100
72   2  2  |GOTO EXIT;                                                             |00012200
73   2  2  |END;                                                                   |00012300
```

```
74   2  1  |IF PDEPTH=0 THEN IF C1='*' | C1='/' THEN DO;                                  |00012400
75   2  2  |PTR1=TERM(SUBSTR(FCTN,1,I-1));                                                |00012500
76   2  2  |PTR2=EXPR(SUBSTR(FCTN,I+1+(SUBSTR(FCTN,I+1,1)='**')));                        |00012600
77   2  2  |RETURN(CODEGEN(PTR1,PTR2,4+(SUBSTR(FCTN,I+1,1)='**')*2-(C1='**')));           |00012700
78   2  2  |END;                                                                          |00012800

79   2  1  |END;  /* OF LOOP DO I = 1 TO LENGTH(FCTN) */                                 |00012900

80   2  0  |IF PDEPTH¬=0 THEN DO;                                                         |00013000
81   2  1  |CALL EMSG('TERM','FEW');                                                      |00013100
82   2  1  |GOTO EXIT;                                                                    |00013200
83   2  1  |END;                                                                          |00013300

84   2  0  |IF C1=')' THEN IF SUBSTR(FCTN,1,1)='(' THEN RETURN(EXPR(SUBSTR(FCTN,2,       |00013400
           |LENGTH(FCTN)-2)));                                                           |00013500

85   2  0  |ELSE DO;  /* PROBABLY A FUNCTION REFERENCE OF FORM CCC(PARM) */             |00013600
86   2  1  |I=INDEX(FCTN,'(');                                                            |00013700
87   2  1  |PTR1=EXPR(SUBSTR(FCTN,I+1,LENGTH(FCTN)-I-1));                                |00013800
88   2  1  |DO J=LBOUND(FUNCS,1) TO HBOUND(FUNCS,1);                                     |00013900
89   2  2  |IF FUNCS(J)=SUBSTR(FCTN,1,I-1) THEN RETURN(CODEGEN(PTR1,NULL,J));            |00014000
90   2  2  |END;                                                                          |00014100
91   2  1  |CALL ERRMES('NAME '||SUBSTR(FCTN,I+1,LENGTH(FCTN)-I-1)||                     |00014200
           |' IS NOT THAT OF A RECOGNIZED FUNCTION');                                    |00014300
92   2  1  |END;                                                                          |00014400

93   2  0  |ELSE DO I=LBOUND(SYMTB.NAME,1) TO HBOUND(SYMTB.NAME,1);                       |00014500
94   2  1  |IF FCTN=SYMTB.NAME(I) THEN DO;                                                |00014600
95   2  2  |AVARNO=I;                                                                     |00014700
96   2  2  |RETURN(SYMTB.PTR(I));                                                         |00014800
97   2  2  |END;                                                                          |00014900
98   2  1  |END;                                                                          |00015000

99   2  0  |IF INDEX('0123456789.''',SUBSTR(FCTN,1,1))=0 THEN GOTO NOCONST;               |00015100

100  2  0  |ON CONVERSION BEGIN;                                                          |00015200
101  3  0  |CALL ERRMES('STRING "'||FCTN||'" IS NOT CONVERTABLE TO A PL/1 NUMERIC C|00015300
           |ONSTANT');                                                                   |00015400
102  3  0  |GOTO EXIT;                                                                    |00015500
103  3  0  |END;                                                                          |00015600

104  2  0  |NUM_CONSTS=NUM_CONSTS+1;                                                      |00015700
105  2  0  |IF NUM_CONSTS>HBOUND(CONSTANT,1) THEN DO;                                     |00015800
106  2  1  |CALL ERRMES('THE INTERNAL TABLE THAT HOLDS NUMERICAL CONSTANTS HAS FILL|00015900
           |ED UP');                                                                     |00016000
107  2  1  |GOTO EXIT;                                                                    |00016100
108  2  1  |END;                                                                          |00016200
109  2  0  |CONSTANT(NUM_CONSTS)=FCTN;  /* CONVERT TO NUMERIC CONSTANT */                |00016300

           |/* THE FOLLOWING CODE PREVENTS SAVING MORE THAN ONE COPY OF EACH CON */     |00016400
110  2  0  |DO I=1 TO NUM_CONSTS-1;                                                       |00016500
111  2  1  |IF CONSTANT(I)=CONSTANT(NUM_CONSTS) THEN DO;                                  |00016600
112  2  2  |NUM_CONSTS=NUM_CONSTS-1; /* RELEASE NEW COPY OF CONSTANT. */                 |00016700
113  2  2  |RETURN(ADDR(CONSTANT(I)));                                                    |00016800
114  2  2  |END;                                                                          |00016900
115  2  1  |END;                                                                          |00017000

           |/* OTHERWISE, IT REALLY IS A NEW CONSTANT, SO SAVE IT. */                    |00017100
116  2  0  |RETURN(ADDR(CONSTANT(NUM_CONSTS)));                                          |00017200

117  2  0  |NOCONST:  CALL ERRMES('"'||FCTN||'" IS NOT A RECOGNIZED VARIABLE');          |00017300
118  2  0  |GOTO EXIT;                                                                    |00017400

119  2  0  |END TERM;                                                                     |00017500

120  1  0  |I=INDEX(FCTN,'=');                                                            |00017600
121  1  0  |IF I>0 THEN DO;                                                               |00017700
122  1  1  |PTR1=EXPR(SUBSTR(FCTN,I+1));                                                  |00017800
123  1  1  |NUM_STEPS=NUM_STEPS+1;                                                        |00017900
124  1  1  |STATEMENT.OPERATION(NUM_STEPS)=0;  /* ASSIGN TO VARIABLE. */                 |00018000
125  1  1  |STATEMENT.OPERAND_ONE(NUM_STEPS)=PTR1;                                        |00018100
126  1  1  |DO J=LBOUND(SYMTB.NAME,1) TO HBOUND(SYMTB.NAME,1);                            |00018200
127  1  2  |IF SYMTB(J).NAME=SUBSTR(FCTN,1,I-1) THEN DO;                                  |00018300
128  1  3  |STATEMENT.RESULT(NUM_STEPS)=SYMTB(J).PTR;                                     |00018400
129  1  3  |AVARNO=J;                                                                     |00018500

           |/* BEFORE RETURNING, MAKE COMPILED BLOCK AS SMALL AS POSSIBLE. */           |00018600
130  1  3  |SIZE=STATEMENT.NUM_STEPS;                                                     |00018700
131  1  3  |ALLOCATE STATEMENT SET(PTR1);                                                 |00018800
132  1  3  |#ELTS=SIZE;  /* IT WAS 50, BEFORE---THIS IS IN OLD ALLOCATION */            |00018900
133  1  3  |PTR1->STATEMENT=STATEMENT; /* OK SINCE BOTH ARE NOW THE "SAME SIZE" */        |00019000
134  1  3  |#ELTS=50;  /* SINCETHE CORRECT VALUE MUST BE RESTORED BEFORE FREEING */     |00019100
135  1  3  |PSPT->BPTR=PTR1;  /* SET POINTER TO CHAIN AROUND ONE WE'RE FREEING */       |00019200
136  1  3  |FREE STATEMENT; /* GET RID OF OLD ONE, WITH SIZE OF 50. */                   |00019300
137  1  3  |RETURN(PTR1);  /* RETURN POINTER TO NEW ALLOCATION. */                        |00019400
138  1  3  |END;                                                                          |00019500
139  1  2  |END;                                                                          |00019600
```

```
140   1  1  |CALL ERRMES('"'||SUBSTR(FCTN,1,I-1)||'" IS NOT A RECOGNIZED VARIABLE');|00019700
141   1  1  |GOTO EXIT;                                                              |00019800
142   1  1  |DCOMPIL:  ENTRY;                                                        |00019900
143   1  1  |PTR1=THREAD_BASE;                                                       |00020000
144   1  1  |DO WHILE(PTR1¬=NULL);                                                   |00020100
145   1  2  |PTR2=PTR1->THREAD;                                                      |00020200
146   1  2  |FREE PTR1->STATEMENT;                                                   |00020300
147   1  2  |PTR1=PTR2;                                                              |00020400
148   1  2  |END;                                                                    |00020500
149   1  1  |THREAD_BASE=NULL;                                                       |00020600
150   1  1  |RETURN;                                                                 |00020700

151   1  1  |VALUATE:  ENTRY(PT) RETURNS(FLOAT BIN);                                 |00020800
152   1  1  |DCL PERFORM(0:10) LABEL,                                                |00020900
           |PT POINTER,                                                             |00021000
           |NUMBER FLOAT BIN BASED,                                                 |00021100
           |(PTR1, PTR2, PTR3) POINTER;                                             |00021200
153   1  1  |IF COMPTST THEN PUT SKIP;                                               |00021300
154   1  1  |DO I=1 TO PT->NUM_STEPS;                                                |00021400
155   1  2  |PTR1=PT->CODE.OPERAND_ONE(I);                                           |00021500
156   1  2  |PTR2=PT->CODE.OPERAND_TWO(I);                                           |00021600
157   1  2  |PTR3=PT->CODE.RESULT(I);                                                |00021700
158   1  2  |GOTO PERFORM(PT->CODE.OPERATION(I));                                    |00021800
159   1  2  |PERFORM(1):  PTR3->NUMBER=PTR1->NUMBER+PTR2->NUMBER;  GOTO ELOOP;       |00021900
161   1  2  |PERFORM(2):  PTR3->NUMBER=PTR1->NUMBER-PTR2->NUMBER;  GOTO ELOOP;       |00022000
163   1  2  |PERFORM(3):  PTR3->NUMBER=PTR1->NUMBER*PTR2->NUMBER;  GOTO ELOOP;       |00022100
165   1  2  |PERFORM(4):  PTR3->NUMBER=PTR1->NUMBER/PTR2->NUMBER;  GOTO ELOOP;       |00022200
167   1  2  |PERFORM(5):  PTR3->NUMBER=PTR1->NUMBER**PTR2->NUMBER;  GOTO ELOOP;      |00022300
169   1  2  |PERFORM(6):  PTR3->NUMBER=SIN (PTR1->NUMBER);  GOTO ELOOP;              |00022400
171   1  2  |PERFORM(7):  PTR3->NUMBER=SIND(PTR1->NUMBER);  GOTO ELOOP;              |00022500
173   1  2  |PERFORM(8):  PTR3->NUMBER=COS (PTR1->NUMBER);  GOTO ELOOP;              |00022600
175   1  2  |PERFORM(9):  PTR3->NUMBER=COSD(PTR1->NUMBER);  GOTO ELOOP;              |00022700
177   1  2  |PERFORM(10):  PTR3->NUMBER=SQRT(PTR1->NUMBER);  GOTO ELOOP;            |00022800
179   1  2  |PERFORM(0):  PTR3->NUMBER=PTR1->NUMBER;                                 |00022900

180   1  2  |ELOOP:                                                                  |00023000
           |IF COMPTST THEN PUT EDIT('* COMPTST -- VALUATE TRACE * '                |00023100
           |||PTR1->NUMBER||' '||                                                  |00023200
           |SUBSTR('   ASSN  +   -   *   /  ** SIN SINDCOS COSDSQRT',             |00023300
           |(PT->CODE.OPERATION(I)+1)*4,4)||' '||PTR2->NUMBER||                     |00023400
           |' -- RESULT IS '||PTR3->NUMBER)(SKIP,A);                               |00023500
181   1  2  |END;                                                                    |00023600
182   1  1  |RETURN(PTR3->NUMBER);                                                   |00023700

183   1  1  |END COMPILE;                                                            |00023800
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	#ELTS	/* IN STATEMENT */ BASED ALIGNED BINARY FIXED (15,0) 4,9,13,13,13,131,132,133,133,133,133,133,133,133,133,133,134,136,136, 136,136,146,146,146,146 52
********	ADDR	BUILTIN 5,10,61,62,66,113,116
2	AVARNO	/* PARAMETER */ ALIGNED BINARY FIXED (15,0) 129
2	BPTR	BASED ALIGNED POINTER 135
2	CODE	(*)/* IN STATEMENT */ BASED /* STRUCTURE */
49	CODEGEN	ENTRY RETURNS(POINTER) 36,77,89
1	COMPILE	EXTERNAL ENTRY RETURNS(POINTER)
2	COMPTST	STATIC EXTERNAL UNALIGNED BIT (1) 153,180
2	CONSTANT	(50) /* IN CONSTANTS */ STATIC ALIGNED INITIAL BINARY /* SINGLE */ FLOAT (21) 1,66,105,109,111,111,113,116
2	CONSTANTS	STATIC /* STRUCTURE */
********	COS	BUILTIN 173
********	COSD	BUILTIN 175
55	C1	AUTOMATIC UNALIGNED CHARACTER (1) 68,69,69,74,74,77,84
25	C1	AUTOMATIC UNALIGNED CHARACTER (1) 27,28,28,33,33,36
2	C1	AUTOMATIC UNALIGNED CHARACTER (1)
142	DCOMPIL	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
2	DCOMPIL	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
180	ELOOP	/* STATEMENT LABEL CONSTANT */ 160,162,164,166,168,170,172,174,176,178
45	EMSG	ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 30,40,71,81
2	ERRMES	EXTERNAL ENTRY(UNALIGNED CHARACTER(*)) RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 140 47,53,91,106,117 101
2	EXIT	STATIC EXTERNAL ALIGNED LABEL 141 31,41,54,72,82,107,118 102
24	EXPR	ENTRY RETURNS(POINTER) 122 35,76,84,87
2	FCTN	AUTOMATIC UNALIGNED CHARACTER (250) VARYING 15,16,16,18,18,18,19,21,21,21,21,22,120,122,127,140
55	FCTN	/* PARAMETER */ UNALIGNED CHARACTER (250) VARYING 66,67,68,75,76,76,77,84,84,84,86,87,87,89,91,91,94,99,109,117 101
25	FCTN	/* PARAMETER */ UNALIGNED CHARACTER (250) VARYING 26,27,34,35,43
55	FUNCS	(6:10) STATIC UNALIGNED INITIAL CHARACTER (4) 64,64,64,64,64,88,88,89
2	FUNCTION	/* PARAMETER */ UNALIGNED CHARACTER(*) 15,15

```
********  HBOUND                  BUILTIN
                                  126
                                  52,60,88,93,105

25        I                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                  26,26,27,34,35

55        I                       AUTOMATIC ALIGNED BINARY FIXED (15,0)


                                  67,67,68,75,76,76,77,86,87,87,89,91,91,93,93,94,95,96,110,110,111,113

2         I                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                  16,16,16,18,18,19,20,21,21,21,22,120,121,122,127,140,154,154,155,156,157,
                                  158,180

********  INDEX                   BUILTIN
                                  19,22,120
                                  86,99

2         J                       AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                  20,20,126,126,127,128,129
                                  88,88,89,89

********  LBOUND                  BUILTIN
                                  126
                                  88,93

********  LENGTH                  BUILTIN
                                  16,18,26,67,84,87,91

2         NAME                    (*) /* IN SYMTB */ /* PARAMETER */ UNALIGNED CHARACTER (12)
                                  126,126,127
                                  93,93,94

117       NOCONST                 /* STATEMENT LABEL CONSTANT */
                                  99

2         NULL                    BUILTIN
                                  3,13,144,149
                                  89
                                  2

2         NUM_CONSTS              /* IN CONSTANTS */ STATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                  104,104,105,109,110,111,112,112,116

2         NUM_STEPS               /* IN STATEMENT */ BASED ALIGNED BINARY FIXED (15,0)
                                  14,123,123,124,125,128,130,133,133,154
                                  51,51,52,56,57,58,61

2         NUM_TEMPS               /* IN TEMPORARIES */ STATIC ALIGNED BINARY FIXED (15,0)
                                  14,59,59,60,61,62

152       NUMBER                  BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                  159,159,159,161,161,161,163,163,163,165,165,165,167,167,167,169,169,171,171,
                                  173,173,175,175,177,177,179,179,180,180,180,182

50        OPCODE                  /* PARAMETER */ ALIGNED BINARY FIXED (15,0)
                                  56

2         OPERAND_ONE             (* REFER (#ELTS) ) /* IN CODE IN STATEMENT */ BASED ALIGNED POINTER
                                  13,125,133,133,155
                                  57

2         OPERAND_TWO             (* REFER (#ELTS) ) /* IN CODE IN STATEMENT */ BASED ALIGNED POINTER
                                  13,133,133,156
                                  58

2         OPERATION               (* REFER (#ELTS) ) /* IN CODE IN STATEMENT */ BASED ALIGNED BINARY FIXED
                                  (15,0)
                                  124,133,133,158,180
                                  52,56

2         P                       STATIC ALIGNED POINTER
                                  6,9,9,10,11,11,13,13,13,13,13,13,13,14,123,123,124,124,125,125,128,128,130,
                                  132,133,133,133,133,133,133,133,133,133,133,134,136,136,136,136,136
                                  51,51,52,52,52,56,56,57,57,58,58,61,61

25        PDEPTH                  AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                  24,28,28,29,33,39

2         PDEPTH                  AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                  14

55        PDEPTH                  AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                  64,69,69,70,74,80

152       PERFORM                 (0:10) AUTOMATIC ALIGNED INITIAL LABEL
                                  1,1,1,1,1,1,1,1,1,1,158
```

```
2          PSET                 AUTOMATIC ALIGNED POINTER
                                5,10,135

152        PT                   /* PARAMETER */ ALIGNED POINTER
                                154,155,156,157,158,180

2          PTR                  (*) /* IN SYMTB */ /* PARAMETER */ ALIGNED POINTER
                                128
                                96

25         PTR1                 AUTOMATIC ALIGNED POINTER
                                34,36

50         PTR1                 /* PARAMETER */ ALIGNED POINTER
                                57

152        PTR1                 AUTOMATIC ALIGNED POINTER
                                122,125,131,131,133,133,133,133,133,133,133,133,133,133,135,137,143,144,
                                145,146,146,146,146,146,147,155,159,161,163,165,167,169,171,173,175,177,179,

65         PTR1                 AUTOMATIC ALIGNED POINTER
                                75,77,87,89

25         PTR2                 AUTOMATIC ALIGNED POINTER
                                35,36

65         PTR2                 AUTOMATIC ALIGNED POINTER
                                76,77

50         PTR2                 /* PARAMETER */ ALIGNED POINTER
                                58

152        PTR2                 AUTOMATIC ALIGNED POINTER
                                145,147,156,159,161,163,165,167,180

152        PTR3                 AUTOMATIC ALIGNED POINTER
                                157,159,161,163,165,167,169,171,173,175,177,179,180,182

2          RESULT               (* REFER (#ELTS) ) /* IN CODE IN STATEMENT */ BASED ALIGNED POINTER
                                13,128,133,133,157
                                61

2          RVARND               /* PARAMETER */ ALIGNED BINARY FIXED (15,0)
                                14,95

********   SIN                  BUILTIN
                                169

********   SIND                 BUILTIN
                                171

2          SIZE                 AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                1,4,4,9,9,130,131,131,132

********   SQRT                 BUILTIN
                                177

2          STATEMENT            BASED (P) /* STRUCTURE */
                                4,4,9,9,13,13,13,131,131,133,133,136,136,146,146
                                52

********   SUBSTR               BUILTIN
                                15,16,18,21,21,21,122,127,140,180
                                27,34,35,68,75,76,76,77,84,84,87,89,91,99

2          SYMTB                (*)/* PARAMETER */ /* STRUCTURE */

********   SYSPRINT             EXTERNAL FILE PRINT
                                153,180

2          TEMPORARIES          STATIC /* STRUCTURE */

2          TEMPRARY             (50) /* IN TEMPORARIES */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                60,61,62

64         TERM                 ENTRY RETURNS(POINTER )
                                34,43,75

2          THREAD               /* IN STATEMENT */ BASED ALIGNED POINTER
                                9,9,10,11,13,133,133,145

2          THREAD_BASE          STATIC ALIGNED INITIAL POINTER
                                3,4,4,5,6,143,149

46         TOO                  /* PARAMETER */ UNALIGNED CHARACTER (10) VARYING
                                47
```

52	TOOBIG	/* STATEMENT LABEL CONSTANT */ 60
46	TYPE	/* PARAMETER */ UNALIGNED CHARACTER (10) VARYING 47
2	VALUATE	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
151	VALUATE	EXTERNAL ENTRY RETURNS(BINARY /* SINGLE */ FLOAT (21))
2	VALUE	AUTOMATIC ALIGNED POINTER
********	VERIFY	BUILTIN 15,21

```
PL/I OPTIMIZING COMPILER          ERRMES:  PROC(ERROR_STR);

                    SOURCE LISTING

    STMT LEV NT

       1      0  |ERRMES:  PROC(ERROR_STR);                                          |00000100
       2    1 0  |DCL ERROR_STR CHAR(*);                                             |00000200
       3    1 0  |PUT FILE(SYSPRINT) EDIT('*****CONTROL CARD ERROR DETECTED.  '      |00000300
                 |||ERROR_STR||'.',                                                  |00000400
                 |'REMAINING CARDS FOR THIS NETWORK ARE BEING FLUSHED.')(SKIP,A);    |00000500
       4    1 0  |END ERRMES;                                                        |00000600
```

```
PL/I OPTIMIZING COMPILER          ERRMES:  PROC(ERROR_STR);

                                          ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.     IDENTIFIER                    ATTRIBUTES AND REFERENCES

1           ERRMES                        EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2           ERROR_STR                     /* PARAMETER */ UNALIGNED CHARACTER(*)
                                          3

********    SYSPRINT                      EXTERNAL FILE PRINT
                                          3
```

```
PL/I OPTIMIZING COMPILER         1PLUCK:  PROC(FREEADDR,TOP);

    STMT LEV NT

      1      0   |PLUCK:  PROC(FREEADDR,TOP);                                          |00000100
      2    1 0   |IF FREEADDR=TOP THEN RETURN;  /* IF FREEING TOP OF STACK ONLY, NO PROB|00000200
                 |LEM AT ALL--RETURN RIGHT AWAY. */                                    |00000300
      3    1 0   |DCL (FREEADDR,TOP) POINTER,                                          |00000400
                 |NULL BUILTIN,                                                       |00000450
                 |(FBIN BASED(FBASE),TBIN BASED(TBASE)) FIXED BIN(31),                |00000500
                 |(FMUCK,TMUCK) POINTER,                                              |00000600
                 |NEWPTR POINTER BASED(TMUCK),                                        |00000700
                 |PREVPTR POINTER BASED(FMUCK);                                       |00000800
      4    1 0   |FMUCK=FREEADDR;  /* PUT ADDR OF ALLOC TO FREE IN A DESTROYABLE PLACE */|00000900
      5    1 0   |TMUCK=TOP; /* PUT ADDRESS OF TOP ALLOCATION IN DESTROYABLE PLACE. */ |00001000
      6    1 0   |TBASE=ADDR(FMUCK);  /* OVERLAY TBIN ON TMUCK. */                     |00001100
      7    1 0   |FBASE=ADDR(FMUCK);       /* OVERLAY FBIN ON FMUCK */                 |00001200
      8    1 0   |TRY_NEXT_ALLOCATION:  TBIN=TBIN-8;  /*TMUCK->CHAIN-BACK ADDRESS */  |00001300
      9    1 0   |IF FMUCK=NEWPTR THEN GOTO GOT_UPPER_ALLOCATION;                      |00001400
                 |/* IF TRUE, TMUCK->CHAIN-BACK FIELD IN THE ALLOC JUST ABOVE THE ONE WE|00001500
                 |WANT TO FREE. */                                                     |00001600
     10    1 0   |IF NEWPTR=NULL THEN SIGNAL ERROR;                                    |00001700
                 |/* THIS MEANS WE'VE HIT THE END OF THE CHAIN, BUT DIDN'T FIND MATCH. */|00001800
     11    1 0   |TMUCK=NEWPTR;  /* SO NEXT TIME TMUCK POINTS TO PREVIOUS ALLOCATION */|00001900
     12    1 0   |GOTO TRY_NEXT_ALLOCATION;  /* LOOK AT NEXT LOWER ONE AS IF IT WERE AT |00002000
                 |THE TOP OF THE STACK. */                                             |00002100
     13    1 0   |GOT_UPPER_ALLOCATION:  FBIN=FBIN-8;                                  |00002200
                 |/* SO NOW FMUCK POINTS TO CHAIN-BACK POINTER OF ALLOCATION TO BE FREED.|00002300
                 |THEN CHAIN BACK ACROSS THE ALLOCATION THAT IS TO BE FREED.  NEXT,    |00002400
                 |SET CHAIN BACK FIELD TO POINT TO THE TOP OF THE NEW STACK. */        |00002500
     14    1 0   |NEWPTR=PREVPTR;                                                      |00002600
     15    1 0   |PREVPTR=TOP;                                                         |00002700
     16    1 0   |END PLUCK;                                                           |00002800
```

```
PL/I OPTIMIZING COMPILER         1PLUCK:  PROC(FREEADDR,TOP);

                                          ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.     IDENTIFIER                    ATTRIBUTES AND REFERENCES

********    ADDR                          BUILTIN
                                          6,7

********    FBASE                         AUTOMATIC ALIGNED POINTER
                                          7,13,13

3           FBIN                          BASED (FBASE) ALIGNED BINARY FIXED (31,0)
                                          13,13

3           FMUCK                         AUTOMATIC ALIGNED POINTER
                                          4,7,9,14,15

3           FREEADDR                      /* PARAMETER */ ALIGNED POINTER
                                          2,4

13          GOT_UPPER_ALLOCATION          /* STATEMENT LABEL CONSTANT */
                                          9

3           NEWPTR                        BASED (TMUCK) ALIGNED POINTER
                                          9,10,11,14

3           NULL                          BUILTIN
                                          10

1           PLUCK                         EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

3           PREVPTR                       BASED (FMUCK) ALIGNED POINTER
                                          14,15

********    TBASE                         AUTOMATIC ALIGNED POINTER
                                          6,8,8

3           TBIN                          BASED (TBASE) ALIGNED BINARY FIXED (31,0)
                                          8,8

3           TMUCK                         AUTOMATIC ALIGNED POINTER
                                          5,6,9,10,11,11,14

3           TOP                           /* PARAMETER */ ALIGNED POINTER
                                          2,5,15

8           TRY_NEXT_ALLOCATION           /* STATEMENT LABEL CONSTANT */
                                          12
```

```
ASSEMBLER OPTIONS=OS,NUM,ALGN,LIST,STMT,XREF,BATCH,EXTEN,NOESD,NORLD,NODECK,NOLOAD,NOLREF,NORENT,NOTERM,
                  NOTEST,UPLIST,EXTIME=5,UTBUFF=3,INSTSET=1,LINECNT=55,NOUPDATE,NOEXECUTE,SPACE=MAX-8K.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                     1          MACRO                                                          00000100
                                     2 &LABEL   PRGET   &CVNAME,&ELTNO,&BASEADR                                00000200
                                     3 &CVNAME  DXD     F                                                      00000300
                                     4 &LABEL   L       15,4(12)              GET ADDR(PRV) FROM PL/1 TCA      00000400
                                     5          LA      0,0(15)               GET ADDR OF DESIRED PSEUDOREG    00000450
                                     6          ORG     *-2                                                    00000500
                                     7          DC      QL2(&CVNAME)                                           00000600
                                     8          ST      0,&ELTNO+&ELTNO+&ELTNO+&ELTNO-4(&BASEADR)              00000700
                                     9          MEND                                                           00000800
000000                               10 PRSGET  CSECT                                                          00000900
                                     11         ENTRY   GET18PR                                                00001000
000000 5811 0000          00000      12 GET18PR L       1,0(1)                GET A(PTR)                       00001100
000004 5811 0000          00000      13         L       1,0(1)                GET POINTER TO ARRAY OF 18 PTS   00001200
                                     14         PRGET   V1,1,1                                                 00001300
                                     15+V1      DXD     F
000008 58F0 0004          00004      16+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
00000C 410F 0000          00000      17+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
000010 000010E                       18+        ORG     *-2
00000E 0000                          19+        DC      QL2(V1)
000010 5001 0000          00000      20+        ST      0,1+1+1+1-4(1)
                                     21         PRGET   V2,2,1                                                 00001400
                                     22+V2      DXD     F
000014 58F0 0004          00004      23+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000018 410F 0000          00000      24+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
00001C 000001A                       25+        ORG     *-2
00001A 0000                          26+        DC      QL2(V2)
00001C 5001 0004          00004      27+        ST      0,2+2+2+2-4(1)
                                     28         PRGET   V3,3,1                                                 00001500
                                     29+V3      DXD     F
000020 58F0 0004          00004      30+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000024 410F 0000          00000      31+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
000026 0000026                       32+        ORG     *-2
000026 0000                          33+        DC      QL2(V3)
000028 5001 0008          00008      34+        ST      0,3+3+3+3-4(1)
                                     35         PRGET   Q1,4,1                                                 00001600
                                     36+Q1      DXD     F
00002C 58F0 0004          00004      37+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000030 410F 0000          00000      38+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
000032 0000032                       39+        ORG     *-2
000032 0000                          40+        DC      QL2(Q1)
000034 5001 000C          0000C      41+        ST      0,4+4+4+4-4(1)
                                     42         PRGET   Q2,5,1                                                 00001700
                                     43+Q2      DXD     F
000038 58F0 0004          00004      44+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
00003C 410F 0000          00000      45+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
00003E 0000003E                      46+        ORG     *-2
00003E 0000                          47+        DC      QL2(Q2)
000040 5001 0010          00010      48+        ST      0,5+5+5+5-4(1)
                                     49         PRGET   Q3,6,1                                                 00001800
                                     50+Q3      DXD     F
000044 58F0 0004          00004      51+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000048 410F 0000          00000      52+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
00004C 0000004A                      53+        ORG     *-2
00004A 0000                          54+        DC      QL2(Q3)
00004C 5001 0014          00014      55+        ST      0,6+6+6+6-4(1)
                                     56         PRGET   D1,7,1                                                 00001900
                                     57+D1      DXD     F
000050 58F0 0004          00004      58+        L       15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000054 410F 0000          00000      59+        LA      0,0(15) GET ADDR OF DESIRED PSEUDOREG
000056 0000056                       60+        ORG     *-2
000056 0000                          61+        DC      QL2(D1)
000058 5001 0018          00018      62+        ST      0,7+7+7+7-4(1)
                                     63         PRGET   D2,8,1                                                 00002000
                                     64+D2      DXD     F
```

```
00005C 58F0 0004          00004    65+         L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000060 410F 0000          00000    66+         LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
000064 000062                      67+         ORG   *-2
000062 0000                        68+         DC    QL2(D2)
000064 5001 001C          0001C    69+         ST    0,8+8+8+8-4(1)
                                   70          PRGET   D3,9,1                                          00002100
                                   71+D3       DXD   F
000068 58F0 0004          00004    72+         L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
00006C 410F 0000          00000    73+         LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
000070 00006E                      74+         ORG   *-2
00006E 0000                        75+         DC    QL2(D3)
000070 5001 0020          00020    76+         ST    0,9+9+9+9-4(1)
                                   77          PRGET   VDT1,10,1                                       00002200
                                   78+VDT1     DXD   F
000074 58F0 0004          00004    79+         L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000078 410F 0000          00000    80+         LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
00007C 00007A                      81+         ORG   *-2
00007A 0000                        82+         DC    QL2(VDT1)
00007C 5001 0024          00024    83+         ST    0,10+10+10+10-4(1)
                                   84          PRGET   VDT2,11,1                                       00002300
                                   85+VDT2     DXD   F
000080 58F0 0004          00004    86+         L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000084 410F 0000          00000    87+         LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
000088 000086                      88+         ORG   *-2
000086 0000                        89+         DC    QL2(VDT2)
000088 5001 0028          00028    90+         ST    0,11+11+11+11-4(1)
                                   91          PRGET   VDT3,12,1                                       00002400
                                   92+VDT3     DXD   F
00008C 58F0 0004          00004    93+         L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000090 410F 0000          00000    94+         LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
000094 000092                      95+         ORG   *-2
000092 0000                        96+         DC    QL2(VDT3)
000094 5001 002C          0002C    97+         ST    0,12+12+12+12-4(1)
                                   98          PRGET   QDT1,13,1                                       00002500
                                   99+QDT1     DXD   F
000098 58F0 0004          00004    100+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
00009C 410F 0000          00000    101+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000A0 00009E                      102+        ORG   *-2
00009E 0000                        103+        DC    QL2(QDT1)
0000A0 5001 0030          00030    104+        ST    0,13+13+13+13-4(1)
                                   105         PRGET   QDT2,14,1                                       00002600
                                   106+QDT2    DXD   F
0000A4 58F0 0004          00004    107+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000A8 410F 0000          00000    108+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000AC 0000AA                      109+        ORG   *-2
0000AA 0000                        110+        DC    QL2(QDT2)
0000AC 5001 0034          00034    111+        ST    0,14+14+14+14-4(1)
                                   112         PRGET   QDT3,15,1                                       00002700
                                   113+QDT3    DXD   F
0000B0 58F0 0004          00004    114+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000B4 410F 0000          00000    115+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000B8 0000B6                      116+        ORG   *-2
0000B6 0000                        117+        DC    QL2(QDT3)
0000B8 5001 0038          00038    118+        ST    0,15+15+15+15-4(1)
                                   119         PRGET   DDT1,16,1                                       00002800
                                   120+DDT1    DXD   F
0000BC 58F0 0004          00004    121+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000C0 410F 0000          00000    122+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000C4 0000C2                      123+        ORG   *-2
0000C2 0000                        124+        DC    QL2(DDT1)
0000C4 5001 003C          0003C    125+        ST    0,16+16+16+16-4(1)
                                   126         PRGET   DDT2,17,1                                       00002900
                                   127+DDT2    DXD   F
0000C8 58F0 0004          00004    128+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000CC 410F 0000          00000    129+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000D0 0000CE                      130+        ORG   *-2
0000CE 0000                        131+        DC    QL2(DDT2)
0000D0 5001 0040          00040    132+        ST    0,17+17+17+17-4(1)
                                   133         PRGET   DDT3,18,1                                       00003000
                                   134+DDT3    DXD   F
0000D4 58F0 0004          00004    135+        L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000D8 410F 0000          00000    136+        LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000DC 0000DA                      137+        ORG   *-2
0000DA 0000                        138+        DC    QL2(DDT3)
0000DC 5001 0044          00044    139+        ST    0,18+18+18+18-4(1)
0000E0 07FE                        140         BR      14                  RETURN TO PL/1 PROGRAM      00003100
                                   141         ENTRY   GETFBPR                                         00003200
0000E2 5811 0000          00000    142 GETFBPR L       1,0(1)                                          00003300
0000E6 5811 0000          00000    143         L       1,0(1)                                          00003400
                                   144         PRGET   FLOWOCK,1,1         STORE ADDR(PR) INTO WHERE PTR-> 00003500
                                   145+FLOWOCK DXD   F
```

```
0000EA 58F0 0004            00004   146+          L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
0000EE 410F 0000            00000   147+          LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
0000F2 0001F0                       148+          ORG   *-2
0000F0 0000                         149+          DC    QL2(FLOWOCK)
0000F2 5001 0000            00000   150+          ST    0,1+1+1+1-4(1)
0000F6 07FE                         151           BR      14                                    00003550
                                    152           ENTRY   GETTFPR                               00003600
0000F8 5811 0000            00000   153 GETTFPR   L       1,0(1)                                00003700
0000FC 5811 0000            00000   154           L       1,0(1)                                00003800
                                    155           PRGET   TFB,1,1                               00003900
                                    156+TFB       DXD   F
000100 58F0 0004            00004   157+          L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
000104 410F 0000            00000   158+          LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
000108 000106                       159+          ORG   *-2
000106 0000                         160+          DC    QL2(TFB)
000104 5001 0000            00000   161+          ST    0,1+1+1+1-4(1)
00010C 07FE                         162           BR      14                                    00004000
                                    163           ENTRY   GETTGPR                               00004100
00010E 5811 0000            00000   164 GETTGPR   L       1,0(1)                                00004200
000112 5811 0000            00000   165           L       1,0(1)                                00004300
                                    166           PRGET   TFB2,1,1                              00004400
                                    167+TFB2      DXD   F
000116 58F0 0004            00004   168+          L     15,4(12) GET ADDR(PRV) FROM PL/1 TCA
00011A 410F 0000            00000   169+          LA    0,0(15) GET ADDR OF DESIRED PSEUDOREG
00011F 00011C                       170+          ORG   *-2
00011C 0000                         171+          DC    QL2(TFB2)
00011E 5001 0000            00000   172+          ST    0,1+1+1+1-4(1)
000122 07FE                         173           BR      14                                    00004500
                                    174           END                                           00004600
```

CROSS-REFERENCE

SYMBOL LENGTH,VALUE,DEFINITION REFERENCES

```
DDT1  1,0,120 124   DDT2  1,0,127 131   DDT3  1,0,134 138   D1  1,0,57 61   D2  1,0,64 68
D3  1,0,71 75   FLOWOCK  1,0,146 149   GETTFBPR  4,B2,142 141   GETTFPR  4,F8,153 152
GETTGPR  4,10E,164 163   GETTSPR  4,0,12 11   PRSGET  1,0,10   QDT1  1,0,99 103   QDT2  1,0,106 110
QDT3  1,0,113 117   Q1  1,0,36 40   Q2  1,0,43 47   Q3  1,0,50 54   TFB  1,0,156 150   TFB2  1,0,167 171
VDT1  1,0,78 82   VDT2  1,0,85 89   VDT3  1,0,92 96   V1  1,0,15 19   V2  1,0,22 26   V3  1,0,29 33
```

```
PL/I OPTIMIZING COMPILER          EVALU8R:  PROC(PT1, PT2, PT3) REORDER;

                    SOURCE LISTING

    STMT LEV NT

      1      0  |EVALU8R:  PROC(PT1, PT2, PT3) REORDER;                          |00000200

                |/* EVALU8R PERFORMS ALL NUMERICAL COMPUTATIONS FOR A PIPE TRIPLET   */|00000210

                |/* MANY VARIABLES ARE DECLARED HERE, OTHERS DECLARED WHERE USED.    */|00000290

      2    1  0  |DCL (PT1,PT2,PT3,PT4) POINTER, NULL BUILTIN,                    |00000300
                |(DDT1,DDT2,DDT3,D1,D2,D3,QDT1,QDT2,QDT3,Q1,Q2,Q3,VDT1,VDT2,VDT3,V1,V2, |00000400
                |V3)(0:#_ELEMENTS-1) FLOAT BIN CONTROLLED EXTERNAL,              |00000500
                |TI FLOAT BIN STATIC EXTERNAL,                                   |00000600
                |#_ELEMENTS FIXED BIN(15),                                       |00000700
                |FLOWBLOCK(SUBSCR1:SUBSCR2) FLOAT BIN CONTROLLED EXTERNAL,       |00000800
                |DEPTHBLOCK(SUBSCR1:SUBSCR2) FLOAT BIN CONTROLLED EXTERNAL,      |00000900
                |GRAPHFL BIT(1) EXTERNAL,                                        |00001000
                |DEBUG CHAR(20) VAR EXTERNAL,                                    |00001100
                |1 NODE BASED(NODE_PTR),                                         |00001200
                |2 NODE_# FIXED BIN(16),                                         |00001300
                |2 TREE_CHAIN_POINTERS(3) POINTER,                               |00001400
                |2 DOWNSTREAM_PIPE_INFO,                                         |00001500
                |3 PIPE_LENGTH FIXED BIN(31),                                    |00001600
                |(3 PIPE_SLOPE,                                                  |00001700
                |3 PIPE_DIAMETER,                                                |00001800
                |3 PIPE_DROP,                                                    |00001900
                |3 PIPE_ROUGHNESS) FLOAT BIN,                                    |00002000
                |3 FUNCTION_INFO,                                                |00002010
                |4 FUNCPTR POINTER,                                              |00002020
                |4 FUNCVARA FIXED BIN(15),                                       |00002030
                |4 FUNCVARB FIXED BIN(15),                                       |00002040
                |2 NODE_TYPE,                                                    |00002100
                |3 MANHOLE BIT(1) ALIGNED,                                       |00002200
                |3 Y_JUNCTION BIT(1) ALIGNED,                                    |00002300
                |3 INPUT_STATION BIT(1) ALIGNED,                                 |00002400
                |3 ROOT_STATION BIT(1) ALIGNED,                                  |00002500
                |3 IMAGINARY BIT(1) ALIGNED,                                     |00002600
                |3 CRASHED BIT(1) ALIGNED,                                       |00002610
                |2 FLOWBLOCK_POINTER POINTER,                                    |00002700
                |2 YBLOCK_POINTER POINTER,                                       |00002800
                |2 CRITICAL_DEPTH FLOAT BIN,                                     |00002900
                |2 TABLE_PAGE# FIXED DEC(4),                                     |00003000
                |2 DELTA_X FLOAT BIN,                                            |00003100
                |2 DNORM FLOAT BIN,                                              |00003200
                |2 MANHOLE_AREA FLOAT BIN,                                       |00003300
                |PAGENO FIXED DEC(4) EXTERNAL,                                   |00003400
                |PRINTERVAL FIXED BIN(15) INIT(10),                              |00003500

                |SCRCHR CHAR(25) VAR,                                            |00003600
                |LASTALLOC EXTERNAL POINTER,                                     |00003700
                |PSEUDOREGISTER POINTER BASED(PSEUDOBASE),                       |00003800
                |GRAPHER ENTRY((*)FLOAT BIN,(*)FLOAT BIN,(*)FLOAT BIN) EXTERNAL, |00003900
                |PRTFLO ENTRY EXTERNAL,                                          |00003920
                |(PTN#1, PTN#2, PTN#3) POINTER EXTERNAL,                         |00003930
                |T FLOAT BIN EXTERNAL,                                           |00003940
                |GOPARM CHAR(20) VAR EXTERNAL,                                   |00003950
                |PLOTTER ENTRY EXTERNAL,                                         |00003960
                |PRTING ENTRY(POINTER) EXTERNAL,  /* PRINTS INPUT HYDROGRAPHS */ |00003970
                |PRTCHG ENTRY EXTERNAL,           /* PRINTS CALCULATED HYDROGRAPHS */|00003980
                |CURPIPE POINTER EXTERNAL,                                       |00003990
                |(LBFB1,LBFB2,HBFB1,HBFB2) FIXED BIN(31),                        |00004100
                |(I,J,K,L,SUBSCR1,SUBSCR2) FIXED BIN(31),                        |00004200
                |TEMP FLOAT BIN,                                                 |00004300
                |TFB(*) FLOAT BIN CONTROLLED EXTERNAL,                           |00004400
                |TFBP POINTER,                                                   |00004500
                |PRTFB POINTER BASED(TFBP),                                      |00004600
                |TFB2(*) FLOAT BIN CONTROLLED EXTERNAL,                          |00004700
                |TFBP2 POINTER,                                                  |00004800
                |PRTFB2 POINTER BASED(TFBP2),                                    |00004900
                |SETDIAM BIT(1) EXTERNAL,                                        |00005000
                |PLOCK ENTRY(POINTER,POINTER),                                   |00005100
                |LAST_TRIPLET BIT(1),                                            |00005200
                |LAST_DOWNSTREAM_DATA FLOAT INIT(0),                             |00005300
                |VALUATE ENTRY EXTERNAL,                                         |00005310
                |OUTDPTH FLOAT BIN EXTERNAL,                                     |00005312
                |1 OUTVARS EXTERNAL,                                             |00005320
                |(2 FT, 2 FDI, 2 FV, 2 FDE) FLOAT BIN,                           |00005330
                |(DIAMETER,DEPTH) FLOAT BIN;                                     |00005400
```

```
                |/* ALLOCATE AND INITIALIZE THE WORK ARRAYS FOR FIRST UPSTREAM PIPE.  */|00005500
 3   1   0      |#_ELEMENTS=PT1->PIPE_LENGTH/PT1->DELTA_X+1.5;                          |00005600
 4   1   0      |ALLOCATE DDT1,D1,QDT1,Q1,VDT1,V1;                                      |00005700
 5   1   0      |DDT1,D1,QDT1,Q1,VDT1,V1=0;                                             |00005800

                |/* ALLOCATE AND INITIALIZE THE WORK ARRAYS FOR SECOND UPSTREAM PIPE. */|00005890
 6   1   0      |#_ELEMENTS=PT2->PIPE_LENGTH/PT2->DELTA_X+1.5;                          |00005900
 7   1   0      |ALLOCATE DDT2,D2,QDT2,Q2,VDT2,V2;                                      |00006000
 8   1   0      |DDT2,D2,QDT2,Q2,VDT2,V2=0;                                             |00006100

                |/* ALLOCATE AND INITIALIZE THE WORK ARRAYS FOR THE DOWNSTREAM PIPE.  */|00006190
 9   1   0      |#_ELEMENTS=PT3->PIPE_LENGTH/PT3->DELTA_X+1.5;                          |00006200
10   1   0      |ALLOCATE DDT3,D3,QDT3,Q3,VDT3,V3;                                      |00006300
11   1   0      |DDT3,D3,QDT3,Q3,VDT3,V3=0;                                             |00006400


12   1   0      |ON ERROR SNAP BEGIN;                                                   |00006500
                |/* ESTABLISH AN ERROR CATCHING ROUTINE SO THAT IF THE PROGRAM       */|00006510
                |/* CRASHES, THE VALUES OF VARIABLES WILL BE PRINTED OUT.             */|00006520
13   2   0      |ON ERROR SNAP SYSTEM;                                                  |00006600
14   2   0      |DCL                                                                    |00006700
                |FIXBIN FIXED BIN(31)  BASED(FIXBINPTR), DUMBPTR POINTER;               |00006800
15   2   0      |FIXBINPTR=ADDR(DUMBPTR);                                               |00006900
16   2   0      |DUMBPTR=PT1;                                                           |00007000
17   2   0      |PUT DATA;                                                              |00007010
                |/* FOLLOWING IS USEFUL FOR PL/1(F), PRIMARILY.                       |00007020
                |DCL FASTDUMP ENTRY(POINTER) RETURNS(CHAR(120));                        |00007030
                |PUT SKIP LIST('FIRST NODE IS:');                                       |00007100
                |DO FIXBIN=FIXBIN TO FIXBIN+64 BY 32;                                   |00007200
                |PUT SKIP EDIT(FASTDUMP(DUMBPTR))(A);                                   |00007300
                |END;                                                                   |00007400
                |DUMBPTR=PT2;                                                           |00007500
                |PUT SKIP LIST('SECOND NODE IS:');                                      |00007600
                |DO FIXBIN=FIXBIN TO FIXBIN+64 BY 32;                                   |00007700
                |PUT SKIP EDIT(FASTDUMP(DUMBPTR))(A);                                   |00007800
                |END;                                                                   |00007900
                |DUMBPTR=PT3;                                                           |00008000
                |PUT SKIP LIST('AND THE THIRD NODE IS:');                               |00008100
                |DO FIXBIN=FIXBIN TO FIXBIN+64 BY 32;                                   |00008200
                |PUT SKIP EDIT(FASTDUMP(DUMBPTR))(A);                                   |00008300
                |END;                                                                   |00008400
                |PUT DATA;                                                              |00008500
                |                                                                   */ |00008510
18   2   0      |END;                                                                   |00008600


                |/* CHECK FOR SOME OF THE VALID "GOPARM" OPTIONS.                     */|00008690

19   1   0      |IF INDEX(GOPARM,'INTVL')>0 THEN DO;                                    |00008700
20   1   1      |SCRCHR=SUBSTR(GOPARM,INDEX(GOPARM,'INTVL'));                           |00008800
21   1   1      |SCRCHR=SUBSTR(SCRCHR,VERIFY(SCRCHR,'INTVL= '));                        |00008900
22   1   1      |GET STRING(SCRCHR) LIST(PRINTERVAL);                                   |00009000
23   1   1      |END;                                                                   |00009100


24   1   0      |PT4=PT3->TREE_CHAIN_POINTERS(1);  /* LOOK AT DOWNSTREAM NODE. */       |00009200
25   1   0      |LAST_TRIPLET=PT4->ROOT_STATION;                                        |00009300
26   1   0      |PTN#1=PT1;  PTN#2=PT2;  PTN#3=PT3;                                     |00009310
                |/* NEXT:  SET UP THE ADDRESSES OF 18 PSEUDOREGISTERS FOR CONTROLLED  |00010100
                |WORK ARRAYS INTO AN 18 ELEMENT ARRAY.  THIS WILL MAKE IT SIMPLE TO     |00010200
                |EXCHANGE THE ARRAYS FOR EACH NEW T+DELTA T.                            |00010300
                |THE "GET" ROUTINES ARE PASSED THE ADDRESS OF A 'PSEUDOREGISTER         |00010400
                |POINTER' AND THEY PUT INTO IT THE ADDRESS OF SOME PSEUDOREGISTER. */   |00010500
29   1   0      |DCL WORKPRS(18) POINTER;                                               |00011000
30   1   0      |DCL (GETFPPR,GETFBPR,GET18PR,GETTGPR) ENTRY(POINTER) EXTERNAL;         |00011100
31   1   0      |CALL GET18PR(ADDR(WORKPRS));                                           |00012800
                |/* ALL OF THE WORK ARRAYS ARE NOW ALLOCATED, AND POINTERS TO THEIR   |00012900
                |PSEUDOREGISTERS ARE SET UP AS WILL BE REQUIRED LATER. */               |00013000
32   1   0      |CALL GETFBPR(ADDR(PSEUDOBASE));                                        |00013100
                |/* PSEUDOREGISTER IS NOW PR(FLOWBLOCK). */                           |00013200
33   1   0      |CALL GETFPR(ADDR(FPBP));                                               |00013210
34   1   0      |CALL GETTGPR(ADDR(FPBP2));                                             |00013220
35   1   0      |NODE_PTR=PT1;       /* SO ON A PUT DATA NODE REALLY IS ONE. */         |00013230
36   1   0      |ALLOCATE QDT_ROOT; QDT_ROOT.ABSTIME=-1.23456; QDT_ROOT.QDTROOT=9.87654;|00013300
39   1   0      |IF PT1->FLOWBLOCK_POINTER=NULL THEN SIGNAL ERROR;                      |00013400
40   1   0      |IF PT2->FLOWBLOCK_POINTER=NULL THEN SIGNAL ERROR;                      |00013500
41   1   0      |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                 |00013600
42   1   0      |IF LBOUND(FLOWBLOCK,1)=-2 THEN IF                                      |00013700
                |HBOUND(FLOWBLOCK,1)=-1 THEN IF                                         |00013800
                |PT2->IMAGINARY='0'B THEN SIGNAL ERROR;                                 |00013900
43   1   0      |ELSE DO;  /* RE-CREATE THE FLOWBLOCK. */                              |00014000
44   1   1      |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                 |00014100
45   1   1      |IF PSEUDOREGISTER=NULL THEN SIGNAL ERROR;                              |00014200
46   1   1      |I=LBOUND(FLOWBLOCK,1);                                                 |00014300
```

```
47   1  1  |J=HBOUND(FLOWBLOCK,1);                                                          |00014400
48   1  1  |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                         |00014500
49   1  1  |CALL PLUCK(PT2->FLOWBLOCK_POINTER,LASTALLOC);                                  |00014600
50   1  1  |FREE FLOWBLOCK;                                                                |00014700
51   1  1  |ALLOCATE FLOWBLOCK(I:J);                                                       |00014800
52   1  1  |LASTALLOC, PT2->FLOWBLOCK_POINTER=PSEUDOREGISTER;                              |00014900
53   1  1  |FLOWBLOCK=0;                                                                   |00015000
54   1  1  |END;                                                                           |00015100

           |/* IF THE SETDIAM BIT IS ON, THE PIPE DIAMETERS ARE TO BE COMPUTED.  */       |00015190

55   1  0  |IF SETDIAM THEN BEGIN;                                                        |00015200
56   2  0  |DCL (MAX1,MAX2,MAX3,GUESS) FLOAT BIN,                                         |00015300
           |TEMPPTR POINTER,                                                              |00015400
           |DIAMSET ENTRY(POINTER) RETURNS(FLOAT BIN) INTERNAL,                           |00015500
           |FINDIAM ENTRY(POINTER, FLOAT BIN) RETURNS(FLOAT BIN);                         |00015600

57   2  0  |DIAMSET:  PROC(POINTER) RETURNS(FLOAT BIN);                                   |00015700
           |/* THIS SUBROUTINE IS GIVEN A POINTER TO A NODE.  IT FINDS THE OPTIMAL       |00015800
           |  DIAMETER AND THEN MAY OR MAY NOT CHANGE THE PIPE DIAMETER. */               |00015900
58   3  0  |DCL POINTER POINTER,                                                          |00016000
           |MAXI FLOAT BIN INIT(-1E5),                                                    |00016100
           |(OPTIMAL_DIAMETER, OLD_DIAM) FLOAT BIN,                                       |00016200
           |HOLD POINTER;                                                                 |00016300
59   3  0  |HOLD=PSEUDOREGISTER;                                                          |00016400
60   3  0  |PSEUDOREGISTER=POINTER->FLOWBLOCK_POINTER;                                    |00016500
61   3  0  |IF PSEUDOREGISTER=NULL THEN SIGNAL ERROR;                                     |00016600
62   3  0  |DO I=LBOUND(FLOWBLOCK,1) TO HBOUND(FLOWBLOCK,1);                              |00016700
63   3  1  |MAXI=MAX(FLOWBLOCK(I),MAXI);  END;                                            |00016800
65   3  0  |OPTIMAL_DIAMETER=FINDIAM(POINTER,MAXI);                                       |00016900
66   3  0  |TEMPPTR=POINTER->TREE_CHAIN_POINTERS(1);                                      |00017000
67   3  0  |OLD_DIAM=POINTER->PIPE_DIAMETER;                                              |00017100
68   3  0  |IF POINTER->CRASHED & OPTIMAL_DIAMETER<OLD_DIAM THEN GOTO RETURN_POINT;       |00017150
69   3  0  |POINTER->PIPE_DIAMETER=OPTIMAL_DIAMETER;                                      |00017200
70   3  0  |IF OPTIMAL_DIAMETER=OLD_DIAM THEN PUT SKIP(2) EDIT                            |00017300
           |('*  DESIGN  *  DIAMETER OF PIPE FROM NODE #',POINTER->NODE_#,                |00017400
           |' TO NODE #',TEMPPTR->NODE_#,' IS ALREADY CORRECT.')                          |00017500
           |(A,2(P'ZZ,ZZ9',A));                                                           |00017600
71   3  0  |ELSE                                                                          |00017700
           |PUT SKIP(2) EDIT('*  DESIGN  *  DIAMETER OF PIPE FROM NODE #',               |00017800
           |POINTER->NODE_#,' TO NODE #', TEMPPTR->NODE_#, ' IS CHANGED FROM ',           |00017900
           |OLD_DIAM,' FEET TO ',OPTIMAL_DIAMETER,' FEET.','IT ',                         |00018000
           |'NEEDS TO BE LARGER','CAN BE SMALLER',' THAN THE ORIGINAL ESTIMATE.')        |00018100
           |(A,2(P'ZZ,ZZ9',A),2(F(6,2),A),SKIP,A,A(18*(OPTIMAL_DIAMETER>OLD_DIAM)),      |00018200
           |A(14*(OPTIMAL_DIAMETER<OLD_DIAM)),A);                                         |00018300
72   3  0  |RETURN_POINT:  PSEUDOREGISTER=HOLD;                                           |00018400
73   3  0  |RETURN(MAXI);                                                                 |00018500
74   3  0  |END DIAMSET;                                                                  |00018600

75   2  0  |FINDIAM:  PROC(NPTR,MAXFLOW) RETURNS(FLOAT BIN);                              |00018700
76   3  0  |DCL NPTR POINTER,                                                             |00018800
           |PDIAMS(0:35) FLOAT BIN STATIC EXTERNAL  INIT(                                 |00018900
           |.5,.5,.75,1,1.25,1.5,1.75,2.0,2.25,2.5,2.75,3.0,3.25,3.5,3.75,4,4.25,        |00019000
           |4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,10,11,12,14,16,18,20,22,24),                    |00019100
           |(MAXFLOW, TESTDIAM, FLOW_RATE_FOR_DIAMETER) FLOAT BIN,                        |00019200
           |(CHOP, LLIM, RLIM, I) FIXED BIN(15);                                          |00019300
77   3  0  |LLIM=LBOUND(PDIAMS,1)-1;                                                      |00019400
78   3  0  |RLIM=HBOUND(PDIAMS,1)+1;                                                      |00019500
79   3  0  |DO I=1 TO 20;                                                                 |00019600
80   3  1  |CHOP=(LLIM+RLIM)/2;                                                           |00019700
81   3  1  |TESTDIAM=PDIAMS(CHOP);                                                        |00019800
82   3  1  |FLOW_RATE_FOR_DIAMETER=6.3*SQRT(NPTR->PIPE_SLOPE*TESTDIAM**5)                 |00019900
           |/* * 1.2 A CORRECTION FACTOR TO GET A SMALLER PIPE, NOW REMOVED. */          |00020000
           |*(2*LOG10(TESTDIAM/(2*NPTR->PIPE_ROUGHNESS))+1.74);                          |00020100
83   3  1  |IF FLOW_RATE_FOR_DIAMETER<=MAXFLOW THEN LLIM=CHOP;                            |00020200
84   3  1  |ELSE RLIM=CHOP+1;                                                             |00020300
85   3  1  |IF CHOP-LLIM=1 THEN RETURN(PDIAMS(CHOP-1));                                   |00020400
86   3  1  |IF RLIM-LLIM=1 THEN RETURN(PDIAMS(RLIM-1));                                   |00020500
87   3  1  |END;                                                                          |00020600
88   3  0  |PUT SKIP(2) LIST('*FINDIAM:  BINARY CHOP FAILS.  DIAMETER SET TO 1.');        |00020700
89   3  0  |RETURN(1);                                                                    |00020800
90   3  0  |END FINDIAM;                                                                  |00020900
```

```
 91   2  0  |MAX1=DIAMSET(PT1);                                                          |00021000
 92   2  0  |IF PT2->IMAGINARY='0'B THEN MAX2=DIAMSET(PT2);                              |00021100

                |/*  THE FOLLOWING SECTION OF CODE GETS AN ESTIMATED PEAK FLOW RATE   */|00021200
                |/*  FOR THE JUNCTION FLOWBLOCK.  THIS IS TO BE USED IN GETTING AN    */|00021300
                |/*  INITIAL ESTIMATE FOR THE DIAMETER NEEDED FOR THE DOWNSTREAM PIPE.*/|00021400
                |/*  THE FIGURE IS BASED ON THE FLOWBLOCKS FOR THE TWO UPSTREAM NODES,*/|00021500
                |/*  AN ESTIMATED TRAVEL TIME THROUGH EACH PIPE ("LAG"), AND THE      */|00021600
                |/*  JUNCTION INFLOW HYDROGRAPH. ("YBLOCK").                          */|00021700

 93   2  0  |MAXFLOW=-1;                                                                 |00021800
 94   2  0  |DCL MAXFLOW FLOAT BIN, LAG FIXED BIN(31), LAG2 FIXED BIN(31);               |00021803
 95   2  0  |LAG1=PT1->PIPE_LENGTH/((SQRT(64.4*PT1->PIPE_DIAMETER*PT1->PIPE_SLOPE)*     |00021806
                |(2*LOG10(PT1->PIPE_DIAMETER/(2*PT1->PIPE_ROUGHNESS))+1.74))*PI);        |00021809
 96   2  0  |IF PT2->IMAGINARY='1'B THEN LAG2=1;  ELSE                                   |00021811
 97   2  0  |LAG2=PT2->PIPE_LENGTH/((SQRT(64.4*PT2->PIPE_DIAMETER*PT1->PIPE_SLOPE)*     |00021812
                |(2*LOG10(PT2->PIPE_DIAMETER/(2*PT2->PIPE_ROUGHNESS))+1.74))*PI);        |00021815
 98   2  0  |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                      |00021818
 99   2  0  |PRTFB=PT2->FLOWBLOCK_POINTER;                                               |00021821
100   2  0  |PRTFB2=PT3->YBLOCK_POINTER;                                                 |00021824
101   2  0  |IF PT3->YBLOCK_POINTER¬=NULL THEN                                           |00021827
                |/* COMPUTE PEAK FLOW RATE WITH A JUNCTION INFLOW HYDROGRAPH PRESENT  */|00021828
                |DO I=MIN(LBOUND(FLOWBLOCK,1),LBOUND(TFB,1),LBOUND(TFB2,1)) TO         |00021831
                |MAX(HBOUND(FLOWBLOCK,1)+LAG1, HBOUND(TFB,1)+LAG2,HBOUND(TFB2,1));     |00021833
102   2  1  |MAXFLOW=MAX(MAXFLOW,                                                        |00021836
                |FLOWBLOCK(MIN(MAX(LBOUND(FLOWBLOCK,1),I-LAG1),HBOUND(FLOWBLOCK,1)))  |00021839
                |+TFB(MIN(MAX(LBOUND(TFB,1),I-LAG2),HBOUND(TFB,1)))                    |00021842
                |+TFB2(MIN(MAX(LBOUND(TFB2,1),I),HBOUND(TFB2,1))));                    |00021845
103   2  1  |END;                                                                        |00021848
104   2  0  |ELSE DO I=MIN(LBOUND(FLOWBLOCK,1),LBOUND(TFB,1)) TO                        |00021853
                |MAX(HBOUND(FLOWBLOCK,1)+LAG1, HBOUND(TFB,1)+LAG2);                    |00021856
                |/* COMPUTE PEAK FLOW RATE WITHOUT ANY JUNCTION INFLOW HYDROGRAPH.    */|00021857
105   2  1  |MAXFLOW=MAX(MAXFLOW,                                                        |00021859
                |FLOWBLOCK(MIN(MAX(LBOUND(FLOWBLOCK,1),I-LAG1),HBOUND(FLOWBLOCK,1)))  |00021862
                |+TFB(MIN(MAX(LBOUND(TFB,1),I-LAG2),HBOUND(TFB,1))));                  |00021865
106   2  1  |IF INDEX(DEBUG,'D')>0 THEN PUT SKIP DATA(I,MAXFLOW);                       |00021866
107   2  1  |END;                                                                        |00021868


108   2  0  |IF PT3->PIPE_DIAMETER¬=0 THEN DO;                                           |00021900
109   2  1  |PUT EDIT('* ESTIMATE *  PHASE BYPASSED FOR THIRD PIPE.  A NON-ZERO DIAM|00022000
                |ETER WAS SPECIFIED ON THE "PD" CARD.')(SKIP,A);                       |00022100
110   2  1  |GOTO ENDBLOCK;                                                              |00022200
111   2  1  |END;                                                                        |00022300

112   2  0  |GUESS=FINDIAM(PT3, MAXFLOW);                                                |00022400

113   2  0  |IF INDEX(DEBUG,'D')>0 THEN PUT SKIP(2) DATA(GUESS);                         |00022410
114   2  0  |PUT EDIT('* ESTIMATE *  AN INITIAL ESTIMATE FOR DIAMETER OF THE PIPE FR|00022600
                |OM NODE #',PT3->NODE_#,' TO NODE #',PT4->NODE_#,' IS ',               |00022700
                |GUESS,' FEET.')                                                       |00022800
                |(SKIP(2),A,2(P'ZZ,ZZ9',A),F(6,2),A);                                  |00022900
115   2  0  |PT3->PIPE_DIAMETER=GUESS;                                                   |00023000
116   2  0  |ENDBLOCK:  END;                                                             |00023100


117   1  0  |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                      |00023200
118   1  0  |LBFB1=LBOUND(FLOWBLOCK,1); HBFB1=HBOUND(FLOWBLOCK,1);                       |00023300
120   1  0  |TEMP=FLOWBLOCK(LBFB1);                                                      |00023400
121   1  0  |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                      |00023500
122   1  0  |LBFB2=LBOUND(FLOWBLOCK,1);  HBFB2=HBOUND(FLOWBLOCK,1);                      |00023600
124   1  0  |TEMP=TEMP+FLOWBLOCK(LBFB2);                                                 |00023700
125   1  0  |SUBSCR1=MIN(LBFB1,LBFB2);                                                   |00023800
126   1  0  |I=HBFB1;  J=HBFB2;                                                          |00023900
128   1  0  |IF I>=J THEN DO;                                                            |00024000
129   1  1  |K=LBFB1;                                                                    |00024100
130   1  1  |SUBSCR2=(I-K)*6+K;  END;                                                    |00024200
132   1  0  |ELSE DO;  K=LBFB2;  SUBSCR2=(J-K)*6+K;  END;                                |00024300
136   1  0  |IF PT3->YBLOCK_POINTER¬=NULL THEN BEGIN;                                    |00024400
137   2  0  |DCL SAVEPTR POINTER;                                                        |00024500
138   2  0  |SAVEPTR=PSEUDOREGISTER;                                                     |00024600
139   2  0  |PSEUDOREGISTER=PT1->YBLOCK_POINTER;                                         |00024700
140   2  0  |CALL PRTIHG(PT3);        /* PRINT THE YBLOCK INPUT HYDROGRAPH. */           |00024800
141   2  0  |TEMP=TEMP+FLOWBLOCK(LBOUND(FLOWBLOCK,1));                                   |00024900
142   2  0  |SUBSCR1=MIN(SUBSCR1,LBOUND(FLOWBLOCK,1));                                   |00025000
143   2  0  |SUBSCR2=MAX(SUBSCR2,HBOUND(FLOWBLOCK,1));                                   |00025100
144   2  0  |PSEUDOREGISTER=SAVEPTR;                                                     |00025200
145   2  0  |END;                                                                        |00025300
                |/* AT THIS POINT SUBSCR1 AND SUBSCR2 HAVE LBOUND(NEW FLOWBLOCK), AND  |00025400
                |HBOUND(NEW FLOWBLOCK). */                                             |00025500
                |/*                                                                    |00025600
                |PUT EDIT('*ON ENTRY, FLOWBLOCK FOR FIRST NODE GOES FROM ',            |00025700
                |LBFB1,' TO ',HBFB1,'.',                                               |00025800
```

```
                |'FOR SECOND NODE GOES FROM ',LBFB2,' TO ',HBFB2,'.',                    |00025900
                |'*THE NEW JUNCTION NODE FLOWBLOCK GOES FROM ',SUBSCR1,' TO ',           |00026000
                |SUBSCR2,'.')                                                            |00026100
                |(SKIP(2),                                                               |00026200
                |A,P'ZZ,ZZ9',A,P'ZZ,ZZ9',A,COL(22),A,P'ZZ,ZZ9',A,P'ZZ,ZZ9',             |00026300
                |A,SKIP,A,P'ZZ,ZZ9',A,P'ZZ,ZZ9',A);                                      |00026400
                |*/                                                                      |00026500
                |/* FIRST, A VALIDITY CHECK BEFORE ALLOCATING THEM:  */                   |00026600
146   1   0     |IF SUBSCR1>LBFB1|SUBSCR1>LBFB2 THEN SIGNAL ERROR;                        |00026700
147   1   0     |IF SUBSCR2<HBFB1|SUBSCR2<HBFB2 THEN SIGNAL ERROR;                        |00026800
148   1   0     |PSEUDOREGISTER=LASTALLOC;                                               |00026900
149   1   0     |ALLOCATE FLOWBLOCK;                                                     |00027000
150   1   0     |FLOWBLOCK=0;                                                            |00027100
151   1   0     |ALLOCATE DEPTHBLOCK;  DEPTHBLOCK=0;                                     |00027200
153   1   0     |IF PT3->FLOWBLOCK_POINTER¬=NULL THEN SIGNAL ERROR;                      |00027300
154   1   0     |LASTALLOC,PT3->FLOWBLOCK_POINTER=PSEUDOREGISTER;                        |00027400
                |/* THE FLOWBLOCK FOR THE JUNCTION NODE IS NOW ALLOCATED AND POINTED     |00027500
                |TO BY THE JUNCTION NODE. */                                             |00027600
155   1   0     |FLOWBLOCK(SUBSCR1)=TEMP;                                                |00027700
156   1   0     |ALLOCATE FLOWBLOCK;                                                     |00027800
157   1   0     |LASTALLOC=PSEUDOREGISTER;                                               |00027900
158   1   0     |CALL GETTFPR(ADDR(TFBP));                                               |00028400

159   1   0     |CALL GETTGPR(ADDR(TFBP2));                                              |00028500
160   1   0     |PRTFB2=PT2->FLOWBLOCK_POINTER;      /* PREVENTS ADDRESSING ON PUT DATA*/|00028600
161   1   0     |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;  /* FLOWBLOCK IS OLD #1 FBLOCK*/|00028700
162   1   0     |IF PT1->INPUT_STATION THEN CALL PRTIHG(PT1); /* PRINT INP HYDROGRAPH */|00028800
163   1   0     |PRTFB=LASTALLOC;  /* TFB IS THE NEW FLOWBLOCK IN NODE #1'S FUTURE. */   |00028900
164   1   0     |DO I=LBFB1 TO HBFB1;                                                    |00029000
165   1   1     |TFB(I)=FLOWBLOCK(I);  END;                                              |00029100
167   1   0     |DO I=LBFB1-1 TO SUBSCR1 BY -1;                                          |00029200
168   1   1     |TFB(I)=TFB(I+1);  END;                                                  |00029300
170   1   0     |DO I=HBFB1+1 TO SUBSCR2;                                                |00029400
171   1   1     |TFB(I)=TFB(I-1);  END;                                                  |00029500
                |/* THE "NEW" FLOWBLOCK FOR NODE #1 IS NOW FULLY INITIALIZED. */         |00029600
173   1   0     |CALL PLUCK(PT1->FLOWBLOCK_POINTER,LASTALLOC);                           |00029700
                |/* PLUCK DE-CHAINS THE ALLOCATION OF THE CONTROLLED VARIABLE WHICH     |00029800
                |WAS AT THE TOP OF THE STACK WHEN ITS PSEUDOREGISTER CONTAINED THE      |00029900
                |QUANTITY IN THE FIRST PARAMETER POINTER.  THE SECOND PARAMETER IS THE  |00030000
                |CONTENTS OF THE CONTROLLED VARIABLES'S PSEUDOREGISTER WHEN THE LAST    |00030100
                |ALLOCATION WAS DONE.  WHEN FOLLOWED BY AN APPROPRIATE SETTING OF THE  |00030200
                |CORRECT PSEUDOREGISTER AND A FREE STATEMENT, THIS ALLOWS FREEING AN    |00030300
                |ALLOCATION FROM THE MIDDLE OF A CONTROLLED VARIABLE STACK, WHILE STILL |00030400
                |MAINTAINING STACK INTEGRITY. */                                         |00030500
174   1   0     |FREE FLOWBLOCK;  /* FLOWBLOCK'S PSEUDOREGISTER ALREADY HAS RIGHT PTR */|00030600
175   1   0     |IF PSEUDOREGISTER¬=LASTALLOC THEN SIGNAL ERROR;                         |00030700
176   1   0     |PSEUDOREGISTER,PT1->FLOWBLOCK_POINTER=LASTALLOC; /* NEW FB FOR NODE 1*/|00030800
177   1   0     |ALLOCATE FLOWBLOCK;  /* WILL BE NEW FLOWBLOCK FOR NODE #2*/             |00030900
178   1   0     |PRTFB,LASTALLOC=PSEUDOREGISTER;                                         |00031000
179   1   0     |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;  /* FLOWBLOCK IS OLD #2 FB */   |00031100
180   1   0     |IF PT2->INPUT_STATION THEN CALL PRTIHG(PT2); /* PRINT INP HYDROGRAPH */|00031200
                |/* NOTE THAT TFB IS THE NEW FLOWBLOCK TO BE USED FOR NODE #2 LATER */   |00031300
181   1   0     |DO I=LBFB2 TO HBFB2;                                                    |00031400
182   1   1     |TFB(I)=FLOWBLOCK(I);  END;                                              |00031500
184   1   0     |DO I=LBFB2-1 TO SUBSCR1 BY -1;                                          |00031600
185   1   1     |TFB(I)=TFB(I+1);  END;                                                  |00031700
187   1   0     |DO I=HBFB2+1 TO SUBSCR2;                                                |00031800
188   1   1     |TFB(I)=TFB(I-1);  END;                                                  |00031900
                |/* NOW THE SECOND NODE'S NEW FLOWBLOCK HAS BEEN FILLED. */             |00032000
190   1   0     |CALL PLUCK(PT2->FLOWBLOCK_POINTER,LASTALLOC); /* PREPARE TO FREE IT */|00032100
191   1   0     |FREE FLOWBLOCK;  /* FREE THE OLD NODE # 2 FLOWBLOCK. */                 |00032200
192   1   0     |IF PSEUDOREGISTER¬=LASTALLOC THEN SIGNAL ERROR;                         |00032300
193   1   0     |PRTFB2,                                                                 |00032400
                |PSEUDOREGISTER,PT2->FLOWBLOCK_POINTER=LASTALLOC;  /* NEW FOR #2. */     |00032500

194   1   0     |RESTART_POINT:                                                          |00032600
                |DT=500000;                                                              |00032610
195   1   0     |CALL DNORMAL(PT1,D1,Q1,V1,QDT1);                                        |00032700
196   1   0     |PT1->DNORM=D1(HBOUND(D1,1));                                            |00032800
197   1   0     |IF PT2->IMAGINARY='0'B THEN                                             |00032900
                |CALL DNORMAL(PT2,D2,Q2,V2,QDT2);                                        |00033000
198   1   0     |PT2->DNORM=D2(HBOUND(D2,1));                                            |00033100
199   1   0     |CALL DNORMAL(PT3,D3,Q3,V3,QDT3);                                        |00033200
200   1   0     |PT3->DNORM=D3(LBOUND(D3,1));                                            |00033300
201   1   0     |DCL DCRIT ENTRY(POINTER) RETURNS(FLOAT BIN);                            |00033400
202   1   0     |PT1->CRITICAL_DEPTH=DCRIT(PT1);                                         |00033500
203   1   0     |IF PT2->IMAGINARY THEN PT2->CRITICAL_DEPTH=0;  ELSE                     |00033600
204   1   0     |PT2->CRITICAL_DEPTH=DCRIT(PT2);                                         |00033700
205   1   0     |PT3->CRITICAL_DEPTH=DCRIT(PT3);                                         |00033800
206   1   0     |IF DT=500000 THEN SIGNAL ERROR;                                         |00033900
207   1   0     |DCL INCOND ENTRY(POINTER,(*) FLOAT BIN,(*) FLOAT BIN);                  |00034000
208   1   0     |DCL CONTROL FLOAT BIN;                                                  |00034010
209   1   0     |IF PT3->DNORM<PT3->CRITICAL_DEPTH THEN DO;                              |00034020
210   1   1     |CONTROL=PT3->CRITICAL_DEPTH;                                            |00034030
```

```
211   1  1  |CALL INCOND(PT3,D3,V3);                                                    |00034040
212   1  1  |END;                                                                       |00034050
213   1  0  |ELSE IF LAST_TRIPLET & PT4->FUNCPTR¬=NULL THEN DO;                         |00034060
214   1  1  |CONTROL=OUTDPTH;                                                           |00034070
215   1  1  |CALL INCOND(PT3,D3,V3);                                                    |00034080
216   1  1  |END;                                                                       |00034090
217   1  0  |ELSE IF PT3->PIPE_DROP>0 THEN DO;                                          |00034100
218   1  1  |CONTROL=PT3->CRITICAL_DEPTH;                                               |00034110
219   1  1  |CALL INCOND(PT3,D3,V3);                                                    |00034120
220   1  1  |END;                                                                       |00034130
221   1  0  |VDT3=V3;   DDT3=D3;                                                        |00034140
223   1  0  |OUTCON=D3(LBOUND(D3,1))+(PT3->MANHOLE='1'B)*V3(LBOUND(V3,1))**2/64.4;      |00034150
224   1  0  |IF PT1->DNORM>PT1->CRITICAL_DEPTH THEN DO;                                 |00034160
225   1  1  |IF PT1->PIPE_DROP+PT1->CRITICAL_DEPTH>OUTCON THEN CONTROL=PT1->            |00034170
            |CRITICAL_DEPTH;                                                           |00034180
226   1  1  |ELSE CONTROL=OUTCON;                                                       |00034190
227   1  1  |CALL INCOND(PT1,D1,V1);                                                    |00034200
228   1  1  |END;                                                                       |00034210
229   1  0  |ELSE DO;                                                                   |00034220
230   1  1  |CONTROL=PT1->CRITICAL_DEPTH;                                               |00034230
231   1  1  |CALL INCOND(PT1,D1,V1);                                                    |00034240
232   1  1  |END;                                                                       |00034250
233   1  0  |IF ¬PT2->IMAGINARY THEN DO;                                                |00034260
234   1  1  |IF PT2->DNORM>PT2->CRITICAL_DEPTH THEN DO;                                 |00034270
235   1  2  |IF PT2->PIPE_DROP + PT2->CRITICAL_DEPTH>OUTCON THEN                        |00034280
            |CONTROL=PT2->CRITICAL_DEPTH;                                              |00034290
236   1  2  |ELSE CONTROL=OUTCON;                                                       |00034300
237   1  2  |CALL INCOND(PT2,D2,V2);                                                    |00034310
238   1  2  |END;                                                                       |00034320
239   1  1  |ELSE DO;                                                                   |00034330

240   1  2  |CONTROL=PT2->CRITICAL_DEPTH;                                               |00034340
241   1  2  |CALL INCOND(PT2,D2,V2);                                                    |00034350
242   1  2  |END;                                                                       |00034360
243   1  1  |END;                                                                       |00034370
244   1  0  |VDT1=V1;  DDT1=D1;  VDT2=V2;  DDT2=D2;                                     |00035300
248   1  0  |IF INDEX(DEBUG,'I')>0 THEN DO;                                             |00035400
249   1  1  |SIGNAL ENDPAGE(SYSPRINT);                                                  |00035500
250   1  1  |PUT LIST('*INITIAL CONDITIONS:  DEBUG="I" WAS SPECIFIED.');                |00035600
251   1  1  |PUT SKIP(2) DATA(D1,D2,D3,V1,V2,V3,Q1,Q2,Q3);                              |00035700
252   1  1  |END;                                                                       |00035800

253   1  0  |BEGIN;                                                                     |00035900
254   2  0  |DCL 1 Q_AND_D_VALUES,                                                      |00036000
            |2 ATIME(2) FLOAT BIN, /* ABSOLUTE TIME, IN SECONDS. */                    |00036100
            |2 QVAL(2) FLOAT BIN,                                                      |00036200
            |2 DVAL(2) FLOAT BIN;                                                      |00036300
255   2  0  |DCL BIT2 BIT(1) INIT('1'B),                                               |00036400
            |UPSTRM ENTRY(POINTER,(*)FLOAT BIN,(*)FLOAT BIN,(*)FLOAT BIN,(*)FLOAT     |00036600
            |BIN, (*) FLOAT BIN),                                                      |00036700
            |NDT /* NEW DT VALUE. */ FLOAT BIN,                                       |00036800
            |INTER ENTRY(POINTER,(*)FLOAT BIN,(*)FLOAT BIN,(*)FLOAT BIN,(*)FLOAT BIN|00036900
            |,(*)FLOAT BIN,(*)FLOAT BIN),                                             |00037000
            |DNSTRM ENTRY,                                                            |00037100
            |BIT BIT(1) EXTERNAL STATIC,                                              |00037200
            |(TPTR1,TPTR2,TPTR3,TPTR4) POINTER,                                       |00037300
            |FLOWINDEX FIXED BIN;                                                     |00037400
256   2  0  |BIT='1'B;                                                                 |00037500
257   2  0  |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                    |00037600
258   2  0  |ATIME(1)=LBOUND(FLOWBLOCK,1)*TI;  /* FIND ABSOLUTE(STARTING) TIME */     |00037700
259   2  0  |QVAL(1)=Q3(LBOUND(FLOWBLOCK,1));                                          |00037800
260   2  0  |DVAL(1)=D3(LBOUND(FLOWBLOCK,1));                                          |00037900
261   2  0  |FLOWINDEX=LBOUND(FLOWBLOCK,1);                                            |00038000
262   2  0  |T=LBOUND(FLOWBLOCK,1)*TI;                                                 |00038100
263   2  0  |CALL PRTFLO;                                                              |00038200

264   2  0  |EX_DO_WHILE_LOOP:  /* BIT MUST BE RESET BY MANHOL OR YJUNCTION IF THEY |00038400
            |DETERMINE THAT ITERATIONS ARE TO CEASE. */                                |00038500
            |T=T+DT;                                                                  |00038600
265   2  0  |NDT=500000;                                                               |00038700
266   2  0  |CALL UPSTRM(PT1,V1,VDT1,D1,DDT1,QDT1);                                    |00038800
267   2  0  |IF ¬PT2->IMAGINARY THEN                                                   |00038900
            |CALL UPSTRM(PT2,V2,VDT2,D2,DDT2,QDT2);                                   |00039000
268   2  0  |CALL INTER(PT1,V1,VDT1,D1,DDT1,QDT1,Q1);                                  |00039100
269   2  0  |IF ¬PT2->IMAGINARY THEN                                                   |00039200
            |CALL INTER(PT2,V2,VDT2,D2,DDT2,QDT2,Q2);                                 |00039300
270   2  0  |IF PT3->MANHOLE | PT3->Y_JUNCTION THEN CALL JUNCTION;  ELSE SIGNAL       |00039400
            |ERROR;                                                                   |00039500
272   2  0  |CALL INTER(PT3,V3,VDT3,D3,DDT3,QDT3,Q3);                                  |00039600
273   2  0  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN                                    |00039700
            |CALL DNSTRM;                                                             |00039800
274   2  0  |FLOWBLOCK_BOUND_EXCEEDED:                                                 |00039900
            |/* NEW LEVEL AT T+DT IS NOW CALCULATED.  NEXT,SAVE NEW Q VALUE IN THE  |00040000
            |FLOWBLOCK.  */                                                           |00040100
            |QVAL(2)=QDT3(LBOUND(QDT3,1));                                            |00040200
```

```
275   2  0  |DVAL(2)=DDT3(LBOUND(DDT3,1));                                                    |00040300
276   2  0  |ATIME(2)=T;                                                                      |00040400
277   2  0  |IF ATIME(1)<0|ATIME(2)<0|QVAL(1)=0|QVAL(2)=0 THEN SIGNAL ERROR;                  |00040500
278   2  0  |FIND_TEMP_FROM_FLOWINDEX:  TEMP=FLOWINDEX*TI;                                    |00040600
279   2  0  |IF TEMP<ATIME(1) THEN SIGNAL ERROR;                                              |00040700
280   2  0  |IF TEMP<=ATIME(2) THEN DO;                                                       |00040800
281   2  1  |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                           |00040900
282   2  1  |ON SUBSCRIPTRANGE BEGIN;                                                         |00041000
283   3  1  |PUT SKIP EDIT('ATTEMPTED TO EXCEED FLOWBLOCK BOUNDS.  ITERATIONS WILL B|00041100
           |E TERMINATED.')(A);                                                              |00041200
284   3  1  |IF FLOWINDEX<LBOUND(FLOWBLOCK,1) THEN SIGNAL ERROR;                              |00041300
285   3  1  |BIT='0'B;                                                                        |00041400
286   3  1  |GOTO FLOWBLOCK_FULL;                                                             |00041500
287   3  1  |END;                                                                             |00041600
288   2  1  |DCL SAVEVAL;                                                                     |00041700
289   2  1  |SAVEVAL=(TEMP-ATIME(1))/(ATIME(2)-ATIME(1));                                     |00041800
290   2  1  |(SUBSCRIPTRANGE):                                                                |00041900
           |FLOWBLOCK(FLOWINDEX)=QVAL(1)+SAVEVAL*(QVAL(2)-QVAL(1));                          |00042000
291   2  1  |REVERT SUBSCRIPTRANGE;                                                           |00042100
292   2  1  |DEPTHBLOCK(FLOWINDEX)=DVAL(1)+SAVEVAL*(DVAL(2)-DVAL(1));                         |00042200
293   2  1  |FLOWINDEX=FLOWINDEX+1;                                                           |00042300
294   2  1  |GOTO FIND_TEMP_FROM_FLOWINDEX;                                                   |00042400
295   2  1  |END;                                                                             |00042500
296   2  0  |ATIME(1)=ATIME(2);                                                               |00042600
297   2  0  |QVAL(1)=QVAL(2);                                                                 |00042700
298   2  0  |DVAL(1)=DVAL(2);                                                                 |00042800
299   2  0  |DVAL(2),                                                                         |00042900
           |QVAL(2),ATIME(2)=0;                                                              |00043000
300   2  0  |DT=NDT;  /* DT IS SET EQUAL TO NEW DT, SET BY SUBROUTINES. */                   |00043100
301   2  0  |DCL DATACOUNT FIXED BIN(31) INIT(0);                                             |00043200

302   2  0  |DATACOUNT=DATACOUNT+1;                                                           |00043300
303   2  0  |IF MOD(DATACOUNT,PRINTERVAL)=0 THEN CALL PRTFLO;                                 |00043400
           |/* NEXT, ALL THE Q<->QDT, D<->DDT, V<->VDT (SWAP ONE FOR ONE) */                 |00046100
304   2  0  |DO I=1 TO 9;                                                                     |00046200
305   2  1  |TPTR1=WORKPRS(I);                                                                |00046300
306   2  1  |TPTR2=WORKPRS(I+9);                                                              |00046400
307   2  1  |TPTR3=TPTR1->PSEUDOREGISTER;                                                     |00046500
308   2  1  |TPTR4=TPTR2->PSEUDOREGISTER;                                                     |00046600
309   2  1  |TPTR1->PSEUDOREGISTER=TPTR4;                                                     |00046700
310   2  1  |TPTR2->PSEUDOREGISTER=TPTR3;                                                     |00046800
311   2  1  |END;                                                                             |00046900
           |/* ALL THE EXCHANGES HAVE NOW BEEN MADE. */                                      |00047000
312   2  0  |IF BIT THEN GOTO EX_DO_WHILE_LOOP;                                               |00047100
           |/* AT THIS POINT FLOWINDEX HAS THE NEXT SUBSCRIPT TO FINISH OUT FLOWB*/           |00047200
313   2  0  |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                           |00047300
314   2  0  |TEMP=FLOWBLOCK(FLOWINDEX-1);                                                     |00047400
315   2  0  |DO FLOWINDEX=FLOWINDEX TO HBOUND(FLOWBLOCK,1);                                   |00047500
316   2  1  |FLOWBLOCK(FLOWINDEX)=TEMP;  END;                                                 |00047600

318   2  0  |(NOSUBSCRIPTRANGE):                                                              |00047700
           |UPSTRM:  PROC(PT,V,VDT,D,DDT,QDT);                                               |00047800
           |/* THIS SUBROUTINE COMPUTES FLOW CONDITIONS AT UPSTREAM NODE STATIONS*/          |00047810
319   3  0  |DCL PT POINTER,                                                                  |00047900
           |TPTR POINTER,                                                                    |00048000
           |(V,VDT,D,DDT,QDT)(*) FLOAT BIN,                                                  |00048100
           |(QINF,REY,RSTR,FUN,FUNPR,DP,TC) FLOAT BIN,                                       |00048200
           |(T1,TS,DS,FUNC,FUNCPR,VS,SF) FLOAT BIN,                                          |00048300
           |(SUB1,SUB2) FIXED BIN,                                                           |00048400
           |/* DEPTH SAME AS CIRCLE. */                                                      |00048500
           |(I,J) FIXED BIN(15);                                                             |00048600
320   3  0  |DCL AK1 FLOAT BIN;                                                               |00048700
321   3  0  |CURPIPE=PT;                                                                      |00048710
322   3  0  |SUB1=T/TI;  /* MAKE SURE THAT T AND TI ARE BOTH INITIALIZED. */                 |00048800
323   3  0  |SUB2=SUB1+1;                                                                     |00048900
324   3  0  |TPTR=PSEUDOREGISTER;                                                             |00049000
325   3  0  |PSEUDOREGISTER=PT->FLOWBLOCK_POINTER;                                            |00049100
326   3  0  |DIAMETER=PT->PIPE_DIAMETER;                                                      |00049200
327   3  0  |ON SUBSCRIPTRANGE BEGIN;                                                         |00049300
           |/* NOTE:  SUBSCRIPTRANGE MIGHT ALSO MEAN THAN WE'RE REFERENCING */              |00049400
           |/* SOME OTHER OUT-OF-BOUNDS ELEMENT IN THE ARRAY. */                            |00049500
328   4  0  |PUT SKIP(3) LIST('*UPSTRM:  FLOWBLOCK FULL.  ITERATIONS TERMINATED.');           |00049600
329   4  0  |BIT='0'B;                                                                        |00049700
330   4  0  |GOTO FLOWBLOCK_BOUND_EXCEEDED;                                                   |00049800
331   4  0  |END;                                                                             |00049900
332   3  0  |(SUBSCRIPTRANGE):                                                                |00050000
           |QINF=FLOWBLOCK(SUB1)+(FLOWBLOCK(SUB2)-FLOWBLOCK(SUB1))*(T-SUB1*TI)/TI;           |00050100
333   3  0  |IF QINF=FLOWBLOCK(SUB1) THEN QINF=.999*FLOWBLOCK(SUB1);                          |00050200
334   3  0  |PSEUDOREGISTER=TPTR;                                                             |00050300
           |/* INTERPOLATES THE VALUE AT TIME T FROM FLOWBLOCK. */                          |00050400
335   3  0  |I=LBOUND(V,1);                                                                   |00050500
336   3  0  |J=LBOUND(D,1);                                                                   |00050600
337   3  0  |IF I^=J THEN SIGNAL ERROR;                                                       |00050700
338   3  0  |T1=DT/PT->DELTA_X;                                                               |00050800
339   3  0  |IF PT->DNORM>PT->CRITICAL_DEPTH THEN DO;                                         |00050900
340   3  1  |TS=T1*(D(I+1)-D(I))/(1+T1*(V(I+1)-V(I)));                                        |00051000
```

```
341   3  1  |DS=D(I);                                                                    |00051100
342   3  1  |FUNC=2;  FUNCPR=1;                                                          |00051200
344   3  1  |DO IT=1 TO 20 WHILE(ABS(FUNC/FUNCPR)>.0005);                                |00051300
345   3  2  |DEPTH=DS;                                                                   |00051400
346   3  2  |CALL CIRCLE;                                                                |00051500
347   3  2  |FUNC=DS-D(I)+TS*(V(I)-C);                                                   |00051600
348   3  2  |FUNCPR=1-TS*(16.1/C)*(1-(2*A*SQRT(DIAMETER*DIAMETER-B*B))/B**3);            |00051700
349   3  2  |DS=DS-FUNC/FUNCPR;                                                          |00051800
350   3  2  |END;                                                                        |00051900
351   3  1  |DEPTH=DS;                                                                   |00052000
352   3  1  |CALL CIRCLE;                                                                |00052100
353   3  1  |F=1/(2*LOG10(2*R/PT->PIPE_ROUGHNESS)+1.74)**2;                              |00052200
354   3  1  |VS=(V(I)+T1*(V(I+1)-V(I))*C)/(1+T1*(V(I+1)-V(I)));                          |00052300

355   3  1  |SF=F*VS*VS/(257.6*R);                                                       |00052400
356   3  1  |CB=VS-AK*DS+32.2*(PT->PIPE_SLOPE-SF)*DT;                                    |00052500
357   3  1  |AK1=AK;                                                                     |00052600
358   3  1  |DP=D(J);                                                                    |00052700
359   3  1  |FUN=2;  FUNPR=1;                                                            |00052800
361   3  1  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                 |00052900
362   3  2  |DEPTH=DP;                                                                   |00053000
363   3  2  |CALL CIRCLE;                                                                |00053100
364   3  2  |FUN=QINF/A-AK1*DP-CB;                                                       |00053200
365   3  2  |FUNPR=-(QINF*B/A**2)-AK1;                                                   |00053400
366   3  2  |DP=DP-FUN/FUNPR;                                                            |00053500
367   3  2  |END;                                                                        |00053600
368   3  1  |IF IIT>=20 THEN PUT SKIP LIST('*UPSTRM:  ITERATIONS EXCEED 20.');           |00053700
369   3  1  |END;                                                                        |00053800
370   3  0  |ELSE DO;                                                                    |00053900
371   3  1  |DEPTH=D(J);                                                                 |00054000
372   3  1  |DP=5000;                                                                    |00054100
373   3  1  |DO IT=1 TO 20 WHILE(ABS(DP-DEPTH)>.0005);                                   |00054200
374   3  2  |IF IT>1 THEN DEPTH=DP;                                                      |00054300
375   3  2  |CALL CIRCLE;                                                                |00054400
376   3  2  |DP=DEPTH-(B*(A**3)-((B*QINF)**2)/32.2)/(3*((B*A)**2)-(2*(A**3)              |00054500
           |* COS(THETA*.5))/SIN(THETA*.5));                                            |00054600
377   3  2  |END;                                                                        |00054700
378   3  1  |IF IT>20 THEN PUT SKIP(2) LIST('UPSTRM:  ITERATIONS(2) EXCEED 20.');        |00054800
379   3  1  |END;                                                                        |00054900
380   3  0  |DDT(J),DEPTH=DP;                                                            |00055000
381   3  0  |QDT(J)=QINF;                                                                |00055100
382   3  0  |CALL CIRCLE;                                                                |00055200
383   3  0  |IF I¬=J THEN SIGNAL ERROR;                                                  |00055300
384   3  0  |VDT(I)=QINF/A;                                                              |00055400
385   3  0  |TC=.9*PT->DELTA_X/ABS(VDT(I)+C);                                            |00055500
386   3  0  |IF TC<NDT THEN NDT=TC;  /* SET NEW DT (NDT) TO MIN(NDT,TC)  */             |00055600
387   3  0  |IF PT->DNORM<PT->CRITICAL_DEPTH THEN DO;                                    |00055700
388   3  1  |DEPTH=D(J+1);                                                               |00055800
389   3  1  |CALL CIRCLE;                                                                |00055900
390   3  1  |F=1/(2*LOG10(2*R/PT->PIPE_ROUGHNESS)+1.74)**2;                              |00056000
391   3  1  |SF=F*V(J+1)**2/(257.6*R);                                                   |00056100
392   3  1  |DV=V(J+2)-V(J);                                                             |00056200
393   3  1  |DY=D(J+2)-D(J);                                                             |00056300
394   3  1  |D2V=V(J+2)-2*V(J+1)+V(J);                                                   |00056400
395   3  1  |D2Y=D(J+2)-2*D(J+1)+D(J);                                                   |00056500
396   3  1  |DDT(J+1)=D(J+1)-.5*T1*(DM*DV+V(J+1)*DY)+.5*T1**2*(2*DM*V(J+1)*D2V+          |00056600
           |(C**2+V(J+1)**2)*D2Y);                                                      |00056700
397   3  1  |VDT(J+1)=V(J+1)-.5*T1*(V(J+1)*DV+32.2*DY+64.4*PT->DELTA_X*(SF-PT->          |00056800
           |PIPE_SLOPE))+.5*T1**2*((V(J+1)**2+C**2)*D2V+64.4*V(J+1)*D2Y);               |00056900
398   3  1  |DEPTH=DDT(J+1);                                                             |00057000
399   3  1  |CALL CIRCLE;                                                                |00057100
400   3  1  |QDT(J+1)=VDT(J+1)*A;                                                        |00057200
401   3  1  |END;                                                                        |00057300

402   3  0  |CURPIPE=NULL;                                                               |00057310
403   3  0  |END UPSTRM;                                                                 |00057400



404   2  0  |(NOSUBSCRIPTRANGE):                                                         |00057500
           |INTER:  PROC(PTR,V,VDT,D,DDT,QDT,Q);                                        |00057600
           |/* THIS SUBROUTINE COMPUTES FLOW CONDITIONS AT INTERIOR STATIONS.    */     |00057610
405   3  0  |DCL (PTR,FPTR) POINTER,                                                     |00057700
           |(V,VDT,D,DDT,QDT,Q) (*) FLOAT BIN,                                          |00057800
           |(I,J,K) FIXED BIN(31),                                                      |00057900
           |L FIXED BIN(31),                                                            |00058000
           |(T1,IB,DR,FUN,FUNPR,AKR,VR,F,SFR) FLOAT BIN,                                |00058100
           |DIAMETER2 FLOAT BIN,                                                        |00058200
           |(TS,DS,AKS,VS,SFS,TC) FLOAT BIN;                                            |00058300
406   3  0  |T1=DT/PTR->DELTA_X;                                                         |00058400
407   3  0  |DIAMETER=PTR->PIPE_DIAMETER;                                                |00058500
408   3  0  |CURPIPE=PTR;                                                                |00058510
409   3  0  |DIAMETER2=DIAMETER**2;                                                      |00058600
410   3  0  |PSEUDOREGISTER=PTR->FLOWBLOCK_POINTER;                                      |00058700
411   3  0  |DO I=LBOUND(V,1)+1+(PTR->DNORM<PTR->CRITICAL_DEPTH)                         |00058800
           | TO HBOUND(V,1)-(PTR->DNORM>=PTR->CRITICAL_DEPTH);                          |00058900
```

```
412   3  1  |J=I-1;  K=I+1;                                                                  |00059000
414   3  1  |IF Q(J)=FLOWBLOCK(LBOUND(FLOWBLOCK,1)) THEN IF Q(J)=Q(HBOUND(Q,1))             |00059100
            |THEN GOTO ENDLOOP;                                                             |00059200
415   3  1  |TR=T1*(D(I)-D(J))/(1+T1*(V(I)-V(J)));                                          |00059300
416   3  1  |DR=D(I);                                                                       |00059400
417   3  1  |FUN=1;  FUNPR=2;                                                               |00059500
419   3  1  |DO L=1 TO 20 WHILE(ABS(FUN/FUNPR)>.001);                                       |00059600
420   3  2  |DEPTH=DR;                                                                      |00059700
421   3  2  |CALL CIRCLE;                                                                   |00059800
422   3  2  |FUN=D(I)-DR-TR*(V(I)+C);                                                       |00059900
423   3  2  |FUNPR=-1-TR*(16.1/C)*(1-(2*A *SQRT(DIAMETER2-B**2))/B**3);                     |00060000
424   3  2  |DR=DR-FUN/FUNPR;                                                               |00060100
425   3  2  |END;                                                                           |00060200
426   3  1  |IF L>=20 THEN PUT SKIP LIST( '*INTER:  ITERATIONS(1) EXCEED 20.');             |00060300
427   3  1  |DEPTH=DR;                                                                      |00060400
428   3  1  |CALL CIRCLE;                                                                   |00060500
429   3  1  |AKR=AK;                                                                        |00060600
430   3  1  |VR=(V(I)-T1*(V(I)-V(J))*C)/(1+T1*(V(I)-V(J)));                                 |00060700
431   3  1  |TVAL=LOG10(2*R/PTR->PIPE_ROUGHNESS);                                           |00060800
432   3  1  |REY=ABS(VR*R/1.2E-5);                                                          |00060810
433   3  1  |RSTR=.633*(TVAL+.87)**8;                                                       |00060820
434   3  1  |IF REY<=RSTR THEN F=.223/REY**.25;                                             |00060830
435   3  1  |ELSE F=1/(2*TVAL+1.74)**2;                                                     |00060840
436   3  1  |SFR=F*ABS(VR)*VR/(257.6*R);                                                    |00060900
437   3  1  |IF PTR->DNORM>PTR->CRITICAL_DEPTH THEN                                         |00061000
            |TS=T1*(D(K)-D(I))/(1+T1*(V(K)-V(I)));                                          |00061100
438   3  1  |ELSE TS=TR;                                                                    |00061200
439   3  1  |DS=D(I);                                                                       |00061300
440   3  1  |FUN=1;                                                                         |00061400
441   3  1  |FUNPR=2;                                                                       |00061500
442   3  1  |DO L=1 TO 20 WHILE(ABS(FUN/FUNPR)>.001);                                       |00061600
443   3  2  |DEPTH=DS;                                                                      |00061700

444   3  2  |CALL CIRCLE;                                                                   |00061800
445   3  2  |FUN=DS-D(I)+TS*(V(I)-C);                                                       |00061900
446   3  2  |FUNPR=1-TS*(16.1/C)*(1-(2*A*SQRT(DIAMETER2-B**2))/B**3);                       |00062000
447   3  2  |DS=DS-FUN/FUNPR;                                                               |00062100
448   3  2  |END;                                                                           |00062200
449   3  1  |IF L>=20 THEN PUT SKIP LIST('*INTER:  ITERATIONS(2) EXCEED 20.');              |00062300
450   3  1  |DEPTH=DS;                                                                      |00062400
451   3  1  |CALL CIRCLE;                                                                   |00062500
452   3  1  |AKS=AK;                                                                        |00062600
453   3  1  |IF PTR->DNORM>PTR->CRITICAL_DEPTH THEN                                         |00062700
            |VS=(V(I)+T1*(V(K)-V(I))*C)/(1+T1*(V(K)-V(I)));                                 |00062800
454   3  1  |ELSE VS=(V(I)+T1*(V(I)-V(J))*C)/(1+T1*(V(I)-V(J)));                            |00062900
455   3  1  |TVAL=LOG10(2*R/PTR->PIPE_ROUGHNESS);                                           |00063000
456   3  1  |REY=ABS(VS*R/1.2E-5);                                                          |00063010
457   3  1  |RSTR=.633*(TVAL+.87)**8;                                                       |00063020
458   3  1  |IF REY<=RSTR THEN F=.223/REY**.25;                                             |00063030
459   3  1  |ELSE F=1/(2*TVAL+1.74)**2;                                                     |00063040
460   3  1  |SFS=F*ABS(VS)*VS/(257.6*R);                                                    |00063100
461   3  1  |DDT(I)=(VR-VS+DR*AKR+DS*AKS+32.2*DT*(SFS-SFR))/(AKR+AKS);                      |00063200
462   3  1  |VDT(I)=VR-AKR*(DDT(I)-DR)-32.2*(SFR-PTR->PIPE_SLOPE)*DT;                       |00063300
463   3  1  |DEPTH=DDT(I);                                                                  |00063400
464   3  1  |CALL CIRCLE;                                                                   |00063500
465   3  1  |QDT(I)= VDT(I)*A;                                                              |00063600
466   3  1  |TC=.9*PTR->DELTA_X/ABS(VDT(I)+C);                                              |00063700
467   3  1  |IF TC<NDT THEN NDT=TC;                                                         |00063800
468   3  1  |ENDLOOP:  END;                                                                 |00063900
469   3  0  |END INTER;                                                                     |00064000


470   2  0  |(NOSUBSCRIPTRANGE):                                                            |00064100
            |DNSTRM:  PROC;                                                                 |00064200
            |/* THIS SUBROUTINE COMPUTES FLOW CONDITIONS AT DOWNSTREAM STATIONS.  */       |00064210
471   3  0  |DCL (NN, I) FIXED BIN(31), (T1, LOGVAL) FLOAT BIN,                             |00064300
            |(RSTR,F,REY) FLOAT BIN;                                                        |00064400
472   3  0  |CURPIPE=PT3;                                                                   |00064410
473   3  0  |DIAMETER= PT3 ->PIPE_DIAMETER;                                                 |00064500
474   3  0  |NN=HBOUND( V3 ,1);  I=NN-1;                                                    |00064600
476   3  0  |SPTR=PT3->TREE_CHAIN_POINTERS(1);       /* POINT TO DOWNSTREAM NODE */         |00064650
477   3  0  |PSEUDOREGISTER= PT3 ->FLOWBLOCK_POINTER;                                       |00064700
478   3  0  |IF Q3(I)=FLOWBLOCK(LBOUND(FLOWBLOCK,1)) THEN IF ¬LAST_TRIPLET (                |00064800
            |PT4->FUNCPTR=NULL THEN RETURN;                                                 |00064801
479   3  0  |DEPTH= D3 (I);                                                                 |00064900
480   3  0  |CALL CIRCLE;                                                                   |00065000
481   3  0  |DMI=DM;                                                                        |00065100
482   3  0  |DEPTH= D3 (NN);                                                                |00065200
483   3  0  |CALL CIRCLE;                                                                   |00065300
484   3  0  |T1=DT/ PT3 ->DELTA_X;                                                          |00065400
485   3  0  |LOGVAL=LOG10(2*R/ PT3 ->PIPE_ROUGHNESS);                                       |00065500
486   3  0  |REY=ABS(V3(NN)*R/1.2E-5);                                                      |00065600
487   3  0  |RSTR=.633*(LOGVAL+.87)**8;                                                     |00065610
488   3  0  |IF REY<=RSTR THEN F=.233/REY**.25;                                             |00065620
489   3  0  |ELSE F=1/(2*LOGVAL+1.74)**2;                                                   |00065630
490   3  0  |SF=F* V3 (NN)**2/(257.6*R);                                                    |00065700
```

```
491   3  0  | DDT3 (NN)= D3 (NN)-T1                                                          |00065800
         |                *(.5*( V3 (NN)+ V3 (I))*( D3 (NN)- D3 (I))+.5*(DMI+DM)*(|00065900
         | V3 (NN)- V3 (I)))                                                              |00066000
         |;                                                                               |00066100
492   3  0  |IF  PT3 ->PIPE_DROP>0 THEN DO;                                                 |00066200
493   3  1  |DEPTH= DDT3 (NN);                                                              |00066300
494   3  1  |CALL CIRCLE;                                                                   |00066400
495   3  1  | VDT3 (NN)=C;     QDT3 (NN)=C*A;     END;                                      |00066500
498   3  0  |ELSE DO;                                                                       |00066600
499   3  1  | VDT3 (NN)= V3 (NN)-T1*( V3 (NN)*( V3 (NN)- V3 (I))+32.2*( D3 (NN)- D3 |00066700
         |(I)))+32.2*DT*                                                                 |00066800
         |( PT3 ->PIPE_SLOPE-SF);                                                        |00066900
500   3  1  |DEPTH= DDT3 (NN);                                                              |00067000
501   3  1  |CALL CIRCLE;                                                                   |00067100
502   3  1  | QDT3 (NN)= VDT3 (NN)*A;                                                       |00067200
503   3  1  |END;                                                                           |00067300

504   3  0  |IF LAST_TRIPLET & PT4->FUNCPTR¬=NULL THEN DO;                                  |00067303
505   3  1  |OUTVARS.PT=T;                                                                  |00067306
506   3  1  |OUTVARS.FDE=DDT3(NN);                                                          |00067308
507   3  1  |OUTVARS.FDI=QDT3(NN);                                                          |00067310
508   3  1  |OUTVARS.FV=VDT3(NN);                                                           |00067312
509   3  1  |F=VALUATE(PT4->FUNCPTR);                                                       |00067314
510   3  1  |IF PT4->FUNCVARR=2 /* = F(Y,DEPTH) */ THEN DO;                                |00067316

511   3  2  |IF PT4->FUNCVARA=3 /* Q,DISCHARGE=F(Y,DEPTH) */ THEN DO;                      |00067318
512   3  3  |QDT3(NN)=OUTVARS.FDI;                                                          |00067320
513   3  3  |VDT3(NN)=OUTVARS.FDI/A;                                                        |00067322
514   3  3  |END;                                                                           |00067324
515   3  2  |ELSE IF PT4->FUNCVARA=4 /* V,VELOCITY = F(Y,DEPTH) */ THEN DO;                |00067326
516   3  3  |VDT3(NN)=OUTVARS.FV;                                                           |00067328
517   3  3  |QDT3(NN)=OUTVARS.FV*A;                                                         |00067330
518   3  3  |END;                                                                           |00067332
519   3  2  |ELSE /* BAD COMBINATION */ ;                                                  |00067334
520   3  2  |END;                                                                           |00067336
521   3  1  |IF PT4->FUNCVARR=1 /* =F(T,TIME) */ | PT4->FUNCVARR=0 THEN DO;                 |00067338
522   3  2  |IF PT4->FUNCVARA=2 /* DEPTH,Y=F(T,TIME) */ THEN DO;                           |00067340
523   3  3  |DDT3(NN)=OUTVARS.FDE;                                                          |00067342
524   3  3  |CALL CIRCLE;                                                                   |00067344
525   3  3  |QDT3(NN)=VDT3(NN)*A;                                                           |00067346
526   3  3  |END;                                                                           |00067348
527   3  2  |ELSE IF PT4->FUNCVARA=3 /* Q,DISCHARGE=F(T,TIME) */ THEN DO;                  |00067350
528   3  3  |QDT3(NN)=OUTVARS.FDI;                                                          |00067352
529   3  3  |VDT3(NN)=QDT3(NN)/A;                                                           |00067354
530   3  3  |END;                                                                           |00067356
531   3  2  |ELSE /* BAD COMBINATION */ ;                                                  |00067358
532   3  2  |END;                                                                           |00067360
533   3  1  |ELSE /* BAD COMBINATION. */ ;                                                 |00067362
534   3  1  |END;                                                                           |00067364
535   3  0  |IF LAST_TRIPLET THEN IF T>LAST_DOWNSTREAM_DATA+TI THEN DO;                     |00067400
536   3  1  |ALLOCATE QDT_ROOT;                                                             |00067500
537   3  1  |QDTROOT= QDT3 (I);                                                             |00067600
538   3  1  |DDTROOT= DDT3 (I);                                                             |00067700
539   3  1  |LAST_DOWNSTREAM_DATA, ABSTIME=T;                                               |00067800
540   3  1  |END;                                                                           |00067900
541   3  0  |CURPIPE=NULL;                                                                  |00067910
542   3  0  |END DNSTRM;                                                                    |00068000

543   2  0  |(NOSUBSCRIPTRANGE):                                                            |00068100
         |JUNCTION:  PROC REORDER;                                                        |00068200
         |/* THIS SUBROUTINE COMPUTES FLOW CONDITIONS AT JUNCTION STATIONS.    */|00068210
544   3  0  |DCL (REY,CP1,CP2,CB3,FUN,FUNPR,RSTR,T1,AK1,AK2,AK3) FLOAT BIN,                |00068300
         |TL FLOAT BIN STATIC EXTERNAL,                                                   |00068400
         |(HB1,HB2,LB3) FIXED BIN(31),                                                    |00068500
         |JUNC_DROP ENTRY(POINTER,(*) FLOAT BIN,(*) FLOAT BIN,(*) FLOAT BIN,             |00068600
         |(*) FLOAT BIN,(*) FLOAT BIN),                                                   |00068700
         |COEFFICIENT ENTRY(FLOAT BIN) INTERNAL;                                          |00068800
545   3  0  |IF PT3->YBLOCK_POINTER=NULL THEN DO;                                           |00068900
546   3  1  |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                         |00069000
547   3  1  |IF Q1(HBOUND(Q1,1)-1)=FLOWBLOCK(LBOUND(FLOWBLOCK,1)) THEN DO;                  |00069100
548   3  2  |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                         |00069200
549   3  2  |IF Q2(HBOUND(Q2,1)-1)=FLOWBLOCK(LBOUND(FLOWBLOCK,1)) THEN RETURN;              |00069300
550   3  2  |END;                                                                           |00069400
551   3  1  |END;                                                                           |00069500
```

```
552   3  0  |COEFFICIENT:  PROC(OUTFLOW_CONDITION);                                  |00069600
            |/* COMPUTE COEFFICIENTS OF THE CHARACTERISTIC EQUATIONS AT JUNCTIONS */|00069610
553   4  0  |IF PT3->MANHOLE & INDEX(DEBUG,'F')>0 THEN PUT SKIP DATA(                |00069615
            |D3(LB3),V3(LB3),D1(HB1),V1(HB1),V2(HB2),D2(HB2)                         |00069620
            |);                                                                     |00069630
554   4  0  |DCL OUTFLOW_CONDITION FLOAT BIN;                                       |00069700
555   4  0  |IF PT1->DNORM>PT1->CRITICAL_DEPTH THEN DO;                             |00069800
556   4  1  |I=HBOUND(V1,1);   J=HBOUND(D1,1);  IF I¬=J THEN SIGNAL ERROR;          |00069900
559   4  1  |DIAMETER=PT1->PIPE_DIAMETER;                                           |00070000
560   4  1  |CURPIPE=PT1;                                                           |00070010
561   4  1  |IF D1(I)+PT1->PIPE_DROP<OUTFLOW_CONDITION THEN DO;                     |00070100
562   4  2  |DEPTH, D1(I)=OUTFLOW_CONDITION-PT1->PIPE_DROP;                         |00070200
563   4  2  |CALL CIRCLE;                                                           |00070400
564   4  2  |V1(I)=Q1(I)/A;                                                         |00070500
565   4  2  |END;                                                                   |00070600
566   4  1  |IF D1(I)<PT1->CRITICAL_DEPTH THEN DO;                                  |00070700
567   4  2  |DEPTH,D1(I)=PT1->CRITICAL_DEPTH;                                       |00070800
568   4  2  |CALL CIRCLE;                                                           |00070900
569   4  2  |V1(I)=C;                                                               |00071000
570   4  2  |Q1(I)=C*A;                                                             |00071100
571   4  2  |END;                                                                   |00071200
572   4  1  |IF D1(I)+PT1->PIPE_DROP=OUTFLOW_CONDITION THEN DO;                     |00071300
573   4  2  |T1=DT/PT1->DELTA_X;                                                    |00071400
574   4  2  |TR=T1*(D1(I)-D1(I-1))/(1+T1*(V1(I)-V1(I-1)));                         |00071500
575   4  2  |DR=D1(I);                                                              |00071600
576   4  2  |FUN=1;                                                                 |00071700
577   4  2  |FUNPR=2;                                                               |00071800
578   4  2  |DO IT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.001);                              |00071900
579   4  3  |DEPTH=DR;                                                              |00072000
580   4  3  |CALL CIRCLE;                                                           |00072100
581   4  3  |FUN=D1(I)-DR-TR*(V1(I)+C);                                             |00072200
582   4  3  |FUNPR=-1-TR*(16.1/C)*(1-(2*A*SQRT(DIAMETER*DIAMETER-B*B))/B**3);       |00072300
583   4  3  |DR=DR-FUN/FUNPR;                                                       |00072400
584   4  3  |END;                                                                   |00072500
585   4  2  |IF IT>20 THEN PUT SKIP LIST('*YJUNCTN:  #1 ITERATIONS EXCEED 20.');    |00072600
586   4  2  |DEPTH=DR;                                                              |00072700
587   4  2  |CALL CIRCLE;                                                           |00072800
588   4  2  |AK1=AK;                                                                |00072900
589   4  2  |VR=(V1(I)-T1*(V1(I)-V1(I-1))*C)/(1+T1*(V1(I)-V1(I-1)));               |00073000
590   4  2  |TVAL=LOG10(2*R/PT1->PIPE_ROUGHNESS);                                   |00073100
591   4  2  |REY=ABS(VR*R/1.2E-5);                                                  |00073110
592   4  2  |RSTR=.633*(TVAL+.87)**8;                                               |00073120
593   4  2  |IF REY<=RSTR THEN F=.223/REY**.25;                                     |00073130
594   4  2  |ELSE F=1/(2*TVAL+1.74)**2;                                             |00073140
595   4  2  |SF=F*ABS(VR)*VR/(257.6*R);                                             |00073200
596   4  2  |CF1=VR+AK*DR+32.2*(PT1->PIPE_SLOPE-SF)*DT;                             |00073300
597   4  2  |END;                                                                   |00073400
598   4  1  |END;                                                                   |00073500
599   4  0  |ELSE IF DESIGN & D3(LBOUND(D3,1))>PT1->PIPE_DIAMETER+PT1->PIPE_DROP    |00073510
600   4  1  |D1(HBOUND(D1,1))=D3(LBOUND(D3,1)); /* WILL CAUSE RESTART IN CIRCLE. */ |00073530
601   4  1  |DEPTH=D1(HBOUND(D1,1));                                                |00073540
602   4  1  |DIAMETER=PT1->PIPE_DIAMETER;                                           |00073550
603   4  1  |CURPIPE=PT1;                                                           |00073560
604   4  1  |CALL CIRCLE;                                                           |00073570
605   4  1  |END;                                                                   |00073580
606   4  0  |IF ¬PT2->IMAGINARY THEN                                                |00073600
            |IF PT2->DNORM>PT2->CRITICAL_DEPTH THEN DO;                             |00073700
607   4  1  |I=HBOUND(V2,1); J=HBOUND(D2,1);  IF I¬=J THEN SIGNAL ERROR;            |00073800
610   4  1  |DIAMETER=PT2->PIPE_DIAMETER;                                           |00073900
611   4  1  |CURPIPE=PT2;                                                           |00073910
612   4  1  |IF D2(I)+PT2->PIPE_DROP<OUTFLOW_CONDITION THEN DO;                     |00074000
613   4  2  |DEPTH, D2(I)=OUTFLOW_CONDITION-PT2->PIPE_DROP;                         |00074100
614   4  2  |CALL CIRCLE;                                                           |00074300
615   4  2  |V2(I)=Q2(I)/A;                                                         |00074400
616   4  2  |END;                                                                   |00074500
617   4  1  |IF D2(I)<PT2->CRITICAL_DEPTH THEN DO;                                  |00074600
618   4  2  |DEPTH,D2(I)=PT2->CRITICAL_DEPTH;                                       |00074700
619   4  2  |CALL CIRCLE;                                                           |00074800
620   4  2  |V2(I)=C;                                                               |00074900
621   4  2  |Q2(I)=A*C;                                                             |00075000
622   4  2  |END;                                                                   |00075100
623   4  1  |IF D2(I)+PT2->PIPE_DROP=OUTFLOW_CONDITION THEN DO;                     |00075200
624   4  2  |T1=DT/PT2->DELTA_X;                                                    |00075300
625   4  2  |TR=T1*(D2(I)-D2(I-1))/(1+T1*(V2(I)-V2(I-1)));                         |00075400
626   4  2  |DR=D2(I);                                                              |00075500
627   4  2  |FUN=1;  FUNPR=2;                                                       |00075600
629   4  2  |DO IT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.001);                              |00075700
630   4  3  |DEPTH=DR;                                                              |00075800
631   4  3  |CALL CIRCLE;                                                           |00075900
632   4  3  |FUN=D2(I)-DR-TR*(V2(I)+C);                                             |00076000
633   4  3  |FUNPR=-1-TR*(16.1/C)*(1-(2*A*SQRT(DIAMETER*DIAMETER-B*B))/(B*B*B));    |00076100
634   4  3  |DR=DR-FUN/FUNPR;                                                       |00076200
635   4  3  |END;                                                                   |00076300
636   4  2  |IF IT>20 THEN PUT SKIP LIST('*YJUNCTN:  #2 ITERATIONS EXCEED 20.');    |00076400
637   4  2  |DEPTH=DR;                                                              |00076500
638   4  2  |CALL CIRCLE;                                                           |00076600
```

```
639  4  2  |AK2=AK;                                                                                    |00076700
640  4  2  |VR=(V2(I)-T1*(V2(I)-V2(I-1))*C)/(1+T1*(V2(I)-V2(I-1)));                                   |00076800
641  4  2  |TVAL=LOG10(2*R/PT2->PIPE_ROUGHNESS);                                                       |00076900
642  4  2  |REY=ABS(VR*R/1.2E-5);                                                                      |00076910
643  4  2  |RSTR=.633*(TVAL+.87)**8;                                                                   |00076920
644  4  2  |IF REY<=RSTR THEN F=.223/REY**.25;                                                         |00076930
645  4  2  |ELSE F=1/(2*TVAL+1.74)**2;                                                                 |00076940
646  4  2  |SF=F*ABS(VR)*VR/(257.6*R);                                                                 |00077000
647  4  2  |CF2=VR+AK*DR+32.2*(PT2->PIPE_SLOPE-SF)*DT;                                                 |00077100
648  4  2  |END;                                                                                       |00077200
649  4  1  |END;                                                                                       |00077300

650  4  0  |ELSE IF DESIGN & D3(LBOUND(D3,1))>PT2->PIPE_DIAMETER+PT2->PIPE_DROP                        |00077310
          |THEN DO;                                                                                  |00077320
651  4  1  |D2(HBOUND(D2,1))=D3(LBOUND(D3,1)); /* WILL CAUSE RESTART IN CIRCLE. */                    |00077330
652  4  1  |DEPTH=D2(HBOUND(D2,1));                                                                    |00077340
653  4  1  |DIAMETER=PT2->PIPE_DIAMETER;                                                               |00077350
654  4  1  |CURPIPE=PT2;                                                                               |00077360
655  4  1  |CALL CIRCLE;                                                                               |00077370
656  4  1  |END;                                                                                       |00077380
657  4  0  |I=LBOUND(V3,1);                                                                            |00077400
658  4  0  |J=LBOUND(D3,1);  IF I¬=J THEN SIGNAL ERROR;                                                |00077500
660  4  0  |DIAMETER=PT3->PIPE_DIAMETER;                                                               |00077600
661  4  0  |CURPIPE=PT3;                                                                               |00077610
662  4  0  |T1=DT/PT3->DELTA_X;                                                                        |00077700
663  4  0  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                                 |00077800
664  4  1  |TR=T1*(D3(I+1)-D3(I))/(1+T1*(V3(I+1)-V3(I)));                                              |00077900
665  4  1  |DS=D3(I);                                                                                  |00078000
666  4  1  |FUN=1;  FUNPR=2;                                                                           |00078100
668  4  1  |DO IT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.001);                                                  |00078200
669  4  2  |DEPTH=DS;                                                                                  |00078300
670  4  2  |CALL CIRCLE;                                                                               |00078400
671  4  2  |FUN=DS-D3(I)+TR*(V3(I)-C);                                                                 |00078500
672  4  2  |FUNPR=1-TR*(16.1/C)*(1-(2*A*SQRT(DIAMETER*DIAMETER-B*B))/B**3);                            |00078600
673  4  2  |DS=DS-FUN/FUNPR;                                                                           |00078700
674  4  2  |END;                                                                                       |00078800
675  4  1  |IF IT>20 THEN PUT SKIP LIST('*YJUNCTN:  #3 ITERATIONS EXCEED 20.');                        |00078900
676  4  1  |DEPTH=DS;                                                                                  |00079000
677  4  1  |CALL CIRCLE;                                                                               |00079100
678  4  1  |AK3=AK;                                                                                    |00079200
679  4  1  |F=1/(2*LOG10(2*R/PT3->PIPE_ROUGHNESS)+1.74)**2;                                            |00079300
680  4  1  |VS=(V3(I)+T1*(V3(I+1)-V3(I))*C)/(1+T1*(V3(I+1)-V3(I)));                                    |00079400
681  4  1  |SF=F*ABS(VS)*VS/(257.6*R);                                                                 |00079500
682  4  1  |CB3=VS-AK*DS+32.2*(PT3->PIPE_SLOPE-SF)*DT;                                                 |00079600
683  4  1  |END;                                                                                       |00079700
684  4  0  |ELSE DO;                                                                                   |00079800
685  4  1  |DV=V3(J+2)-V3(J);                                                                          |00079900
686  4  1  |DY=D3(J+2)-D3(J);                                                                          |00080000
687  4  1  |D2V=V3(J+2)-2*V3(J+1)+V3(J);                                                               |00080100
688  4  1  |D2Y=D3(J+2)-2*D3(J+1)+D3(J);                                                               |00080200
689  4  1  |DEPTH=D3(J+1);                                                                             |00080300
690  4  1  |CALL CIRCLE;                                                                               |00080400
691  4  1  |TVAL=LOG10(2*R/PT3->PIPE_ROUGHNESS);                                                       |00080500
692  4  1  |REY=ABS(V3(J+1)*R/1.2E-5);                                                                 |00080510
693  4  1  |RSTR=.633*(TVAL+.87)**8;                                                                   |00080520
694  4  1  |IF REY<=RSTR THEN F=.223/REY**.25;                                                         |00080530
695  4  1  |ELSE F=1/(2*TVAL+1.74)**2;                                                                 |00080540
696  4  1  |SF=F*V3(J+1)**2/(257.6*R);                                                                 |00080600
697  4  1  |DEPTH,DDT3(J+1)=D3(J+1)-.5*T1*(DM*DV+V3(J+1)*DY)+.5*T1**2*(2*DM*                           |00080700
          |V3(J+1)*D2V+(C**2+V3(J+1)**2)*D2Y);                                                        |00080800
698  4  1  |VDT3(J+1)=V3(J+1)-.5*T1*(V3(J+1)*DV+32.2*DY+64.4*PT3->DELTA_X                              |00080900

          |*(SF-PT3->PIPE_SLOPE))+.5*T1**2*((V3(J+1)**2+C**2)*D2V+64.4*V3(J+1)                        |00081000
          |*D2Y);                                                                                     |00081100
699  4  1  |CALL CIRCLE;                                                                               |00081200
700  4  1  |QDT3(J+1)=VDT3(J+1)*A;                                                                     |00081300
701  4  1  |END;                                                                                       |00081400
702  4  0  |END COEFFICIENT;                                                                           |00081500
703  3  0  |(NOSUBSCRIPTRANGE): /* SUBSCRIPTRANGE ERRORS CANNOT OCCUR IN HERE. */                     |00081600
          |Y_INTERPOLATOR:  PROC(TIME) RETURNS(FLOAT);                                                |00081700
          |/* COMPUTES DIRECT JUNCTION INFLOW BY LINEAR INTERPOLATIONS.          */                    |00081710
704  4  0  |DCL SAVEPTR POINTER, RETVAL FLOAT, TIMEINC FIXED BIN(31);                                  |00081800
705  4  0  |SAVEPTR=PSEUDOREGISTER;                                                                    |00081900
706  4  0  |PSEUDOREGISTER=PT3->YBLOCK_POINTER;                                                        |00082000
707  4  0  |TIMEINC=TIME/TI;                                                                           |00082100
708  4  0  |IF TIMEINC<=LBOUND(FLOWBLOCK,1) THEN RETVAL=FLOWBLOCK(LBOUND(FLOWBLOCK,                   |00082200
          |1));                                                                                       |00082300
709  4  0  |ELSE IF TIMEINC>=HBOUND(FLOWBLOCK,1) THEN RETVAL=FLOWBLOCK(HBOUND(                        |00082400
          |FLOWBLOCK,1));                                                                             |00082500
710  4  0  |ELSE /* WE CAN INTERPOLATE THE VALUE. */                                                   |00082600
          |RETVAL=FLOWBLOCK(TIMEINC)+((T-TIMEINC*TI)/TI)*(FLOWBLOCK(TIMEINC+1)                        |00082700
          |-FLOWBLOCK(TIMEINC));                                                                      |00082800
711  4  0  |PSEUDOREGISTER=SAVEPTR;                                                                    |00082900
712  4  0  |RETURN(RETVAL);                                                                            |00083000
713  4  0  |END Y_INTERPOLATOR;                                                                        |00083100
```

```
714   3  0  |HB1=HBOUND(D1,1);                                                                    |00083200
715   3  0  |HB2=HBOUND(D2,1);                                                                    |00083300
716   3  0  |LB3=LBOUND(D3,1);                                                                    |00083400
717   3  0  |DCL INFLOW FLOAT;                                                                    |00083500
718   3  0  |IF PT2->IMAGINARY THEN QDT2(HB2)=0;                                                  |00083600
719   3  0  |IF PT3->YBLOCK_POINTER¬=NULL THEN INFLOW=Y_INTERPOLATOR(T);                          |00083700
720   3  0  |ELSE INFLOW=0;                                                                       |00083800
721   3  0  |IF PT3->Y_JUNCTION THEN IT_IS_A_Y_JUNCTION:  DO;                                     |00083900
722   3  1  |CALL COEFFICIENT(D3(LB3));                                                           |00084000
           |/* THIS SECTION DOES COMPUTATIONS FOR A POINT-TYPE JUNCTION.                        */|00084010
723   3  1  |IF PT1->PIPE_DROP + D1(HB1)=D3(LB3) THEN                                             |00084100
           |IF PT2->PIPE_DROP+D2(HB2)=D3(LB3) THEN DO;                                           |00084200
724   3  2  |DP=D3(LB3);                                                                          |00084300
725   3  2  |FUN=2;  FUNPR=1;                                                                     |00084400
727   3  2  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                          |00084500
728   3  3  |DEPTH=DP-PT1->PIPE_DROP;                                                             |00084600
729   3  3  |DIAMETER=PT1->PIPE_DIAMETER;                                                         |00084700
730   3  3  |CURPIPE=PT1;                                                                         |00084710
731   3  3  |CALL CIRCLE;                                                                         |00084800
732   3  3  |DCL (AA1,AA2,AA3,BB1,BB2,BB3) FLOAT BIN;                                             |00084900
733   3  3  |AA1=A;                                                                               |00085000
734   3  3  |BB1=B;                                                                               |00085100
735   3  3  |DEPTH=DP-PT2->PIPE_DROP;                                                             |00085200
736   3  3  |DIAMETER=PT2->PIPE_DIAMETER;                                                         |00085300
737   3  3  |CURPIPE=PT2;                                                                         |00085310
738   3  3  |CALL CIRCLE;                                                                         |00085400
739   3  3  |AA2=A;                                                                               |00085500
740   3  3  |BB2=B;                                                                               |00085600
741   3  3  |DEPTH=DP;                                                                            |00085700
742   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                                         |00085800
743   3  3  |CURPIPE=PT3;                                                                         |00085810
744   3  3  |CALL CIRCLE;                                                                         |00085900
745   3  3  |AA3=A;                                                                               |00086000
746   3  3  |BB3=B;                                                                               |00086100
747   3  3  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                           |00086200
748   3  4  |FUN=(CF1-AK1*(DP-PT1->PIPE_DROP))*AA1 + (CF2-AK2*(DP-PT2->PIPE_DROP))*               |00086300
           |AA2-(CB3+AK3*DP)*AA3+INFLOW;                                                         |00086400
749   3  4  |FUNPR=(CF1-AK1*(DP-PT1->PIPE_DROP))*BB1 + (CF2-AK2*(DP-PT2->PIPE_DROP))              |00086500
           |* BB2 - (CB3+AK3*DP)*BB3-AK1*AA1-AK2*AA2-AK3*AA3;                                    |00086600
750   3  4  |END;                                                                                 |00086700
751   3  3  |ELSE DO;                                                                             |00086800
752   3  4  |FUN=(CF1-AK1*(DP-PT1->PIPE_DROP))*AA1 + (CF2-AK2*(DP-PT2->PIPE_DROP))*               |00086900
           |AA2-C*AA3+INFLOW;                                                                    |00087000
753   3  4  |FUNPR=(CF1-AK1*(DP-PT1->PIPE_DROP))*BB1 + (CF2-AK2*(DP-PT2->PIPE_DROP))              |00087100
           |*BB2-C*BB3-AK1*AA1-AK2*AA2-(16.1*AA3/C)*(B**3-2*A*DIAMETER*COS(.5*THETA              |00087200
           |))/B**3;                                                                             |00087300
754   3  4  |END;                                                                                 |00087400
755   3  3  |DP=DP-FUN/FUNPR;                                                                     |00087500
756   3  3  |END;                                                                                 |00087600

757   3  2  |IF IIT>=20 THEN PUT SKIP LIST('*YJUNCTN:  ITERATIONS(1) EXCEED 20.');                |00087700
758   3  2  |DEPTH,DDT1(HB1)=DP-PT1->PIPE_DROP;                                                   |00087800
759   3  2  |VDT1(HB1)=CF1-AK1*DDT1(HB1);                                                         |00087900
760   3  2  |DIAMETER=PT1->PIPE_DIAMETER;                                                         |00088000
761   3  2  |CURPIPE=PT1;                                                                         |00088010
762   3  2  |CALL CIRCLE;                                                                         |00088100
763   3  2  |QDT1(HB1)=VDT1(HB1)*A;                                                               |00088200
764   3  2  |DEPTH,DDT2(HB2)=DP-PT2->PIPE_DROP;                                                   |00088300
765   3  2  |VDT2(HB2)=CF2-AK2*DDT2(HB2);                                                         |00088400
766   3  2  |DIAMETER=PT2->PIPE_DIAMETER;                                                         |00088500
767   3  2  |CURPIPE=PT2;                                                                         |00088510
768   3  2  |CALL CIRCLE;                                                                         |00088600
769   3  2  |QDT2(HB2)=VDT2(HB2)*A;                                                               |00088700
770   3  2  |QDT3(LB3)=QDT1(HB1)+QDT2(HB2)+INFLOW;                                                |00088800
771   3  2  |DEPTH, DDT3(LB3)=DP;                                                                 |00088900
772   3  2  |DIAMETER=PT3->PIPE_DIAMETER;                                                         |00089000
773   3  2  |CURPIPE=PT3;                                                                         |00089010
774   3  2  |CALL CIRCLE;                                                                         |00089100
775   3  2  |VDT3(LB3)=QDT3(LB3)/A;                                                               |00089200
776   3  2  |END;  /* OF "INSERT # 1" */                                                         |00089300
777   3  1  |ELSE IF PT2->PIPE_DROP+D2(HB2)¬=D3(LB3) THEN DO;                                     |00089400
778   3  2  |IF PT2->DNORM>PT2->CRITICAL_DEPTH THEN                                               |00089500
           |CALL JUNC_DROP(PT2,V2,D2,DDT2,VDT2,QDT2);                                            |00089600
779   3  2  |DP=D3(LB3);                                                                          |00089700
780   3  2  |FUN=2;  FUNPR=1;                                                                     |00089800
782   3  2  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                          |00089900
783   3  3  |ITERATION:  DEPTH=DP-PT1->PIPE_DROP;                                                 |00090000
784   3  3  |DIAMETER=PT1->PIPE_DIAMETER;                                                         |00090100
785   3  3  |CURPIPE=PT1;                                                                         |00090110
786   3  3  |CALL CIRCLE;                                                                         |00090200
787   3  3  |AA1=A;                                                                               |00090300
788   3  3  |BB1=B;                                                                               |00090400
789   3  3  |DEPTH=DP;                                                                            |00090500
790   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                                         |00090600
```

```
791   3  3  |CURPIPE=PT3;                                                                    |00090610
792   3  3  |CALL CIRCLE;                                                                    |00090700
793   3  3  |AA3=A;                                                                          |00090800
794   3  3  |BB3=B;                                                                          |00090900
795   3  3  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                      |00091000
796   3  4  |FUN=QDT2(HB2)+(CF1-AK1*(DP-PT1->PIPE_DROP))*AA1-(CB3+AK3*DP)*AA3+               |00091100
            |INFLOW;                                                                        |00091200
797   3  4  |FUNPR=(CF1-AK1*(DP-PT1->PIPE_DROP))*BB1-(CB3+AK3*DP)*BB3-AK1*AA1-AK3           |00091300
            |*AA3;                                                                          |00091400
798   3  4  |END;                                                                            |00091500
799   3  3  |ELSE DO;                                                                        |00091600
800   3  4  |FUN=QDT2(HB2)+(CF1-AK1*(DP-PT1->PIPE_DROP))*AA1-C*AA3+INFLOW;                   |00091700
801   3  4  |FUNPR=(CF1-AK1*(DP-PT1->PIPE_DROP))*BB1-C*BB3-AK1*AA1-(16.1*AA3/C)*            |00091800
            |(B**3-2*A*DIAMETER*COS(.5*THETA))/B**3;                                        |00091900
802   3  4  |END;                                                                            |00092000

803   3  3  |DP=DP-FUN/FUNPR;                                                                |00092100
804   3  3  |END;                                                                            |00092200
805   3  2  |IF IIT>=20 THEN PUT SKIP LIST('**YJUNCTN:  ITERATIONS(2) EXCEED 20.');         |00092300
806   3  2  |DEPTH,DDT1(HB1)=DP-PT1->PIPE_DROP;                                              |00092400
807   3  2  |VDT1(HB1)=CF1-AK1*DDT1(HB1);                                                    |00092500
808   3  2  |DIAMETER=PT1->PIPE_DIAMETER;                                                    |00092600
809   3  2  |CURPIPE=PT1;                                                                    |00092610
810   3  2  |CALL CIRCLE;                                                                    |00092700
811   3  2  |QDT1(HB1)=VDT1(HB1)*A;                                                          |00092800
812   3  2  |DEPTH,DDT3(LB3)=DP;                                                             |00092900
813   3  2  |QDT3(LB3)=QDT2(HB2)+QDT1(HB1)+INFLOW;                                           |00093000
814   3  2  |DIAMETER=PT3->PIPE_DIAMETER;                                                    |00093100
815   3  2  |CURPIPE=PT3;                                                                    |00093110
816   3  2  |CALL CIRCLE;                                                                    |00093200
817   3  2  |VDT3(LB3)=QDT3(LB3)/A;                                                          |00093300
818   3  2  |IF LB3¬=J THEN SIGNAL ERROR;                                                    |00093400
819   3  2  |END; /* OF "INSERT # 4" */                                                     |00093500
820   3  1  |ELSE SIGNAL ERROR;                                                              |00093600
821   3  1  |ELSE IF PT1->PIPE_DROP+D1(HB1)¬=D3(LB3) THEN DO;                                |00093700
822   3  2  |IF PT1->DNORM>PT1->CRITICAL_DEPTH THEN                                          |00093800
            |CALL JUNC_DROP(PT1,V1,D1,DDT1,VDT1,QDT1);                                      |00093900
823   3  2  |IF PT2->PIPE_DROP+D2(HB2)¬=D3(LB3) THEN DO;                                     |00094000
824   3  3  |IF PT2->DNORM>PT2->CRITICAL_DEPTH THEN                                          |00094100
            |CALL JUNC_DROP(PT2,V2,D2,DDT2,VDT2,QDT2);                                      |00094200
825   3  3  |QDT3(J)=QDT1(HB1)+QDT2(HB2)+INFLOW;                                             |00094300
826   3  3  |DP=D3(J);                                                                       |00094400
827   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                                    |00094500
828   3  3  |CURPIPE=PT3;                                                                    |00094510
829   3  3  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                      |00094600
830   3  4  |FUN=2;  FUNPR=1;                                                                |00094700
832   3  4  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                     |00094800
833   3  5  |DEPTH=DP;                                                                       |00094900
834   3  5  |CALL CIRCLE;                                                                    |00095000
835   3  5  |FUN=QDT3(J)/A-AK3*DP-CB3;                                                       |00095100
836   3  5  |FUNPR=-(QDT3(J)*B/A**2)-AK3;                                                    |00095200
837   3  5  |DP=DP-FUN/FUNPR;                                                                |00095300
838   3  5  |END;                                                                            |00095400
839   3  4  |IF IIT>=20 THEN PUT SKIP LIST('**YJUNCTN:  ITERATIONS(3) EXCEED 20.');         |00095500
840   3  4  |END;                                                                            |00095600
841   3  3  |ELSE DO;                                                                        |00095700
842   3  4  |DEPTH=D3(J);                                                                    |00095800
843   3  4  |DP=5000;                                                                        |00095900
844   3  4  |DO IIT=1 TO 20 WHILE(ABS(DP-DEPTH)>.0005);                                      |00096000
845   3  5  |IF IIT>1 THEN DEPTH=DP;                                                         |00096100
846   3  5  |CALL CIRCLE;                                                                    |00096200
847   3  5  |DP=DEPTH-(B*(A**3)-((B*QDT3(J))**2)/32.2)/(3*((B*A)**2)-(2*(A**3)*             |00096300
            |COS(THETA*.5))/SIN(THETA*.5));                                                 |00096400
848   3  5  |END;                                                                            |00096500
849   3  4  |IF IIT>20 THEN PUT SKIP LIST('**YJUNCTN:  ITERATIONS(3.1) EXCEED 20.');        |00096600

850   3  4  |END;                                                                            |00096700
851   3  3  |DEPTH,DDT3(J)=DP;                                                               |00096800
852   3  3  |CALL CIRCLE;                                                                    |00096900
853   3  3  |VDT3(J)=QDT3(J)/A;                                                              |00097000
854   3  3  |END;  /* OF "INSERT # 2" */                                                    |00097100
855   3  2  |ELSE IF PT2->PIPE_DROP+D2(HB2)=D3(LB3) THEN DO;                                 |00097200
856   3  3  |DP=D3(J);                                                                       |00097300
857   3  3  |FUN=2;  FUNPR=1;                                                                |00097400
859   3  3  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                     |00097500
860   3  4  |DEPTH=DP-PT2->PIPE_DROP;                                                        |00097600
861   3  4  |DIAMETER=PT2->PIPE_DIAMETER;                                                    |00097700
862   3  4  |CURPIPE=PT2;                                                                    |00097710
863   3  4  |CALL CIRCLE;                                                                    |00097800
864   3  4  |AA2=A;                                                                          |00097900
865   3  4  |BB2=B;                                                                          |00098000
866   3  4  |DEPTH=DP;                                                                       |00098100
867   3  4  |DIAMETER=PT3->PIPE_DIAMETER;                                                    |00098200
868   3  4  |CURPIPE=PT3;                                                                    |00098210
869   3  4  |CALL CIRCLE;                                                                    |00098300
```

```
870   3  4  |AA3=A;                                                                    |00098400
871   3  4  |BB3=B;                                                                    |00098500
872   3  4  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                |00098600
873   3  5  |FUN=QDT1(HB1)+(CF2-AK2*(DP-PT2->PIPE_DROP))*AA2-(CB3+AK3*DP)*AA3+         |00098700
            |INFLOW;                                                                   |00098800
874   3  5  |FUNPR=(CF2-AK2*(DP-PT2->PIPE_DROP))*BB2-(CB3+AK3*DP)*BB3-AK2*AA2-AK3      |00098900
            |*AA3;                                                                     |00099000
875   3  5  |END;                                                                      |00099100
876   3  4  |ELSE DO;                                                                  |00099200
877   3  5  |FUN=QDT1(HB1)+(CF2-AK2*(DP-PT2->PIPE_DROP))*AA2-C*AA3+INFLOW;             |00099300
878   3  5  |FUNPR=(CF2-AK2*(DP-PT2->PIPE_DROP))*BB2-AK2*AA2-C*BB3-(16.1*AA3/C)*      |00099400
            |(B**3-2*A*DIAMETER*COS(.5*THETA))/B**3;                                   |00099500
879   3  5  |END;                                                                      |00099600
880   3  4  |DP=DP-FUN/FUNPR;                                                          |00099700
881   3  4  |END;                                                                      |00099800
882   3  3  |IF IIT>=20 THEN PUT SKIP LIST('*YJUNCTN:  ITERATIONS(4) EXCEED 20.');     |00099900
883   3  3  |DEPTH,DDT2(HB2)=DP-PT2->PIPE_DROP;                                        |00100000
884   3  3  |VDT2(HB2)=CF2-AK2*DDT2(HB2);                                              |00100100
885   3  3  |DIAMETER=PT2->PIPE_DIAMETER;                                              |00100200
886   3  3  |CURPIPE=PT2;                                                              |00100210
887   3  3  |CALL CIRCLE;                                                              |00100300
888   3  3  |QDT2(HB2)=VDT2(HB2)*A;                                                    |00100400
889   3  3  |QDT3(J)=QDT1(HB1)+QDT2(HB2);                                              |00100500
890   3  3  |DEPTH,DDT3(J)=DP;                                                         |00100600
891   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                              |00100700
892   3  3  |CURPIPE=PT3;                                                              |00100710
893   3  3  |CALL CIRCLE;                                                              |00100800
894   3  3  |VDT3(J)=QDT3(J)/A;                                                        |00100900
895   3  3  |END; /* OF "INSERT # 3" */                                               |00101000
896   3  2  |ELSE SIGNAL ERROR;                                                        |00101100

897   3  2  |END;  /* OF "INSERT # 2FIRST" */                                         |00101200
898   3  1  |ELSE SIGNAL ERROR;                                                        |00101300
899   3  1  |END IT_IS_A_Y_JUNCTION;  /* END OF POINT-TYPE JUNCTION SECTION.      */|00101400

900   3  0  |ELSE IT_IS_A_MANHOLE_JUNCTION:  DO;                                       |00101500
            |/* THIS SECTION DOES COMPUTATIONS FOR A RESERVOIR-TYPE JUNCTION.      */|00101510
901   3  1  |IF INDEX(DEBUG,'F')>0 THEN DO;  PUT SKIP LIST('* IN MANHOLE:');          |00101515
903   3  2  |PUT DATA(D3(LB3),V3(LB3),D1(HB1),V1(HB1),D2(HB2),V2(HB2));               |00101520
904   3  2  |END;                                                                      |00101530
905   3  1  |DCL OUTFLOW_CONDITION FLOAT BIN;                                          |00101600
906   3  1  |OUTFLOW_CONDITION=D3(LB3)+V3(LB3)**2/64.4;                                |00101700
907   3  1  |CALL COEFFICIENT(OUTFLOW_CONDITION);                                      |00101800
908   3  1  |IF INDEX(DEBUG,'F')>0 THEN PUT SKIP DATA(OUTFLOW_CONDITION);              |00101810
909   3  1  |IF ABS(PT1->PIPE_DROP+D1(HB1)-OUTFLOW_CONDITION)<.0001 THEN               |00101900
            |IF ABS(PT2->PIPE_DROP+D2(HB2)-OUTFLOW_CONDITION)<.0001 THEN DO;           |00102000
910   3  2  |DP3=D3(LB3);                                                              |00102100
911   3  2  |FUN=2;  FUNPR=1;                                                          |00102200
913   3  2  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                               |00102300
914   3  3  |DEPTH=DP3;                                                                |00102400
915   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                              |00102500
916   3  3  |CURPIPE=PT3;                                                              |00102600
917   3  3  |CALL CIRCLE;                                                              |00102700
918   3  3  |AA3=A;                                                                    |00102800
919   3  3  |BB3=B;                                                                    |00102900
920   3  3  |VELOCITY3=CB3+AK3*DP3;                                                    |00103000
921   3  3  |DP1=DP3+VELOCITY3**2/64.4-PT1->PIPE_DROP;                                 |00103100
922   3  3  |DIAMETER=PT1->PIPE_DIAMETER;                                              |00103200
923   3  3  |CURPIPE=PT1;                                                              |00103300
924   3  3  |CALL CIRCLE;                                                              |00103400
925   3  3  |AA1=A;                                                                    |00103500
926   3  3  |BB1=B;                                                                    |00103600
927   3  3  |VELOCITY1=CF1-AK1*DP1;                                                    |00103700
928   3  3  |DP2=DP3+VELOCITY3**2/64.4-PT2->PIPE_DROP;                                 |00103800
929   3  3  |DIAMETER=PT2->PIPE_DIAMETER;                                              |00103900
930   3  3  |CURPIPE=PT2;                                                              |00104000
931   3  3  |CALL CIRCLE;                                                              |00104100
932   3  3  |AA2=A;                                                                    |00104200
933   3  3  |BB2=B;                                                                    |00104300
934   3  3  |VELOCITY2=CF2-AK2*DP2;                                                    |00104400
935   3  3  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                |00104500
936   3  4  |FUN=VELOCITY1*AA1+VELOCITY2*AA2-VELOCITY3*AA3-(DP1-D1(HB1))*              |00104600
            |PT3->MANHOLE_AREA/DT+INFLOW;                                              |00104700
937   3  4  |FUNPR=(VELOCITY1*BB1+VELOCITY2*BB2-AK1*AA1-AK2*AA2-PT3->MANHOLE_AREA      |00104800
            |/DT)*(1+VELOCITY3*AK3/32.2)-(VELOCITY3*BB3+AK3*AA3);                      |00104900
938   3  4  |END;                                                                      |00105000
939   3  3  |ELSE DO;                                                                  |00105100
940   3  4  |DEPTH=DP3;                                                                |00105200
941   3  4  |DIAMETER=PT3->PIPE_DIAMETER;                                              |00105300
942   3  4  |CURPIPE=PT3;                                                              |00105400
943   3  4  |CALL CIRCLE;                                                              |00105500
944   3  4  |FUN=VELOCITY1*AA1+VELOCITY2*AA2-C*AA3-(DP1-D1(HB1))*PT3->MANHOLE_AREA     |00105600
            |/DT+INFLOW;                                                               |00105700
945   3  4  |FUNPR=(VELOCITY1*BB1+VELOCITY2*BB2-AK1*AA1-AK2*AA2-PT3->MANHOLE_AREA      |00105800
```

```
                |/DT)*(1+VELOCITY3*AK3/32.2)-C*BB3-(16.1*AA3/C)*(B*B-2*A*DIAMETER*      |00105900
                |COS(.5*THETA))/B**3;                                                  |00106000
946   3  4      |END;                                                                  |00106100
947   3  3      |DP3=DP3-FUN/FUNPR;                                                    |00106200
948   3  3      |END;                                                                  |00106300
949   3  2      |DEPTH,DDT3(LB3)=DP3;                                                  |00106400
950   3  2      |VDT3(LB3)=CB3+AK3*DP3;                                                |00106500
951   3  2      |DIAMETER=PT3->PIPE_DIAMETER;                                          |00106600
952   3  2      |CURPIPE=PT3;                                                          |00106700
953   3  2      |CALL CIRCLE;                                                          |00106800
954   3  2      |QDT3(LB3)=VDT3(LB3)*A;                                                |00106900
955   3  2      |DEPTH,DDT1(HB1)=DP3+VDT3(LB3)**2/64.4-PT1->PIPE_DROP;                 |00107000
956   3  2      |VDT1(HB1)=CF1-AK1*DDT1(HB1);                                          |00107100
957   3  2      |DIAMETER=PT1->PIPE_DIAMETER;                                          |00107200
958   3  2      |CURPIPE=PT1;                                                          |00107300
959   3  2      |CALL CIRCLE;                                                          |00107400
960   3  2      |QDT1(HB1)=VDT1(HB1)*A;                                                |00107500
961   3  2      |DEPTH,DDT2(HB2)=DP3+VDT3(LB3)**2/64.4-PT2->PIPE_DROP;                 |00107600
962   3  2      |VDT2(HB2)=CF2-AK2*DDT2(HB2);                                          |00107700
963   3  2      |DIAMETER=PT2->PIPE_DIAMETER;                                          |00107800
964   3  2      |CURPIPE=PT2;                                                          |00107900
965   3  2      |CALL CIRCLE;                                                          |00108000
966   3  2      |QDT2(HB2)=VDT2(HB2)*A;                                                |00108100
967   3  2      |END;                                                                  |00108200


968   3  1      |ELSE DO;                                                              |00115800
969   3  2      |IF PT2->DNORM>PT2->CRITICAL_DEPTH THEN                                |00115900
                |/* NOTE:  IF PT2->IMAGINARY THEN DNORM==CRITICAL_DEPTH. */           |00116000
                |CALL JUNC_DROP(PT2,V2,D2,DDT2,VDT2,QDT2);                            |00116100
970   3  2      |DP3=D3(LB3);                                                          |00116200
971   3  2      |FUN=2;  FUNPR=1;                                                      |00116300
973   3  2      |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                           |00116400
974   3  3      |DEPTH=DP3;                                                            |00116500
975   3  3      |DIAMETER=PT3->PIPE_DIAMETER;                                          |00116600
976   3  3      |CURPIPE=PT3;                                                          |00116700
977   3  3      |CALL CIRCLE;                                                          |00116800
978   3  3      |AA3=A;                                                                |00116900
979   3  3      |BB3=B;                                                                |00117000
980   3  3      |VELOCITY3=CB3+AK3*DP3;                                                |00117100
981   3  3      |DP1=DP3+VELOCITY3**2/64.4-PT1->PIPE_DROP;                             |00117200
982   3  3      |DIAMETER=PT1->PIPE_DIAMETER;                                          |00117300
983   3  3      |CURPIPE=PT1;                                                          |00117400
984   3  3      |CALL CIRCLE;                                                          |00117500
985   3  3      |AA1=A;                                                                |00117600
986   3  3      |BB1=B;                                                                |00117700
987   3  3      |VELOCITY1=CF1-AK1*DP1;                                                |00117800
988   3  3      |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                            |00117900
989   3  4      |FUN=QDT2(HB2)+VELOCITY1*AA1-VELOCITY3*AA3-(DP1-D1(HB1))*              |00118000

                |PT3->MANHOLE_AREA/DT+INFLOW;                                          |00118100
990   3  4      |FUNPR=(VELOCITY1*BB1-AK1*AA1-PT3->MANHOLE_AREA/DT)*(1+VELOCITY3*AK3   |00118200
                |/32.2)-(VELOCITY3*BB3+AK3*AA3);                                       |00118300
991   3  4      |END;                                                                  |00118400
992   3  3      |ELSE DO;                                                              |00118500
993   3  4      |DEPTH=DP3;                                                            |00118600
994   3  4      |DIAMETER=PT3->PIPE_DIAMETER;                                          |00118700
995   3  4      |CURPIPE=PT3;                                                          |00118800
996   3  4      |CALL CIRCLE;                                                          |00118900
997   3  4      |FUN=QDT2(HB2)+VELOCITY1*AA1-C*AA3-(DP1-D1(HB1))*PT3->MANHOLE_AREA/DT  |00119000
                |+INFLOW;                                                              |00119100
998   3  4      |FUNPR=(VELOCITY1*BB1-AK1*AA1-PT3->MANHOLE_AREA/DT)*(1+VELOCITY3*AK3   |00119200
                |/32.2)-C*BB3-(16.1*AA3/C)*(B*3-2*A*DIAMETER*COS(.5*THETA))/B**3;      |00119300
999   3  4      |END;                                                                  |00119400
1000  3  3      |DP3=DP3-FUN/FUNPR;                                                    |00119500
1001  3  3      |END;                                                                  |00119600
1002  3  2      |DEPTH,DDT3(LB3)=DP3;                                                  |00119700
1003  3  2      |VDT3(LB3)=CB3+AK3*DP3;                                                |00119800
1004  3  2      |DIAMETER=PT3->PIPE_DIAMETER;                                          |00119900
1005  3  2      |CURPIPE=PT3;                                                          |00120000
1006  3  2      |CALL CIRCLE;                                                          |00120100
1007  3  2      |QDT3(LB3)=VDT3(LB3)*A;                                                |00120200
1008  3  2      |DEPTH,DDT1(HB1)=DP3+VDT3(LB3)**2/64.4-PT1->PIPE_DROP;                 |00120300
1009  3  2      |QDT1(HB1)=QDT3(LB3)-QDT2(HB2);                                        |00120400
1010  3  2      |DIAMETER=PT1->PIPE_DIAMETER;                                          |00120500
1011  3  2      |CURPIPE=PT1;                                                          |00120600
1012  3  2      |CALL CIRCLE;                                                          |00120700
1013  3  2      |VDT1(HB1)=QDT1(HB1)/A;                                                |00120800
1014  3  2      |END;                                                                  |00120900
1015  3  1      |ELSE IF ABS(PT1->PIPE_DROP+D1(HB1)-OUTFLOW_CONDITION)>.0001 THEN DO;  |00122000
1016  3  2      |IF PT1->DNORM>PT1->CRITICAL_DEPTH THEN                                |00122100
                |CALL JUNC_DROP(PT1,V1,D1,DDT1,VDT1,QDT1);                            |00122200
1017  3  2      |IF ABS(PT2->PIPE_DROP+D2(HB2)-OUTFLOW_CONDITION)>.0001 THEN DO;       |00122300
1018  3  3      |IF PT2->DNORM>PT1->CRITICAL_DEPTH THEN                                |00122400
                |CALL JUNC_DROP(PT2,V2,D2,DDT2,VDT2,QDT2);                            |00122500
```

```
1019   3  3  |DP3=2*D3(LB3)-DDT3(LB3);                                                                    |00122600
1020   3  3  |E3=D3(LB3)+V3(LB3)**2/64.4;                                                                 |00122700
1021   3  3  |DCL (DISCRG,STORG) FLOAT BIN;                                                               |00122800
1022   3  3  |DISCRG=QDT1(HB1)+QDT2(HB2)+INFLOW;                                                          |00122900
1023   3  3  |STORG=PT3->MANHOLE_AREA/DT;                                                                 |00123000
1024   3  3  |FUN=2;  FUNPR=1;                                                                            |00123100
1026   3  3  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                                 |00123200
1027   3  4  |VELOCITY3=CB3+AK3*DP3;                                                                      |00123300
1028   3  4  |DIAMETER=PT3->PIPE_DIAMETER;                                                                |00123400
1029   3  4  |CURPIPE=PT3;                                                                                |00123500
1030   3  4  |DEPTH=DP3;                                                                                  |00123600
1031   3  4  |CALL CIRCLE;                                                                                |00123700
1032   3  4  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                                  |00123800
1033   3  5  |FUN=DISCRG-VELOCITY3*A-STORG*(DP3+VELOCITY3**2/64.4-E3);                                    |00123900

1034   3  5  |FUNPR=-VELOCITY3*B-AK3*A-STORG*(1+VELOCITY3*AK3/32.2);                                      |00124000
1035   3  5  |END;                                                                                        |00124100
1036   3  4  |ELSE DO;                                                                                    |00124200
1037   3  5  |FUN=DISCRG-C*A-STORG*(DP3+C**2/64.4-E3);                                                    |00124300
1038   3  5  |FUNPR=-(16.1*A/C)*(B**3-2*A* DIAMETER*COS(.5*THETA))/B**3-C*B-                              |00124400
              |STORG*(1.5*B**3-A*DIAMETER*COS(.5*THETA))/B**3;     /* COMMON TERM? */                      |00124500
1039   3  5  |END;                                                                                        |00124600
1040   3  4  |DP3=DP3-FUN/FUNPR;                                                                          |00124700
1041   3  4  |END;                                                                                        |00124800
1042   3  3  |IF IIT>20 THEN PUT SKIP LIST('*JUNCTION:  MANHOLE ITERATIONS(3) EXCEED                      |00124900
              |20.');                                                                                      |00125000
1043   3  3  |DEPTH,DDT3(LB3)=DP3;                                                                        |00125100
1044   3  3  |CURPIPE=PT3;                                                                                |00125200
1045   3  3  |CALL CIRCLE;                                                                                |00125300
1046   3  3  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN                                                      |00125400
              |VDT3(LB3)=CB3+AK3*DP3;                                                                      |00125500
1047   3  3  |ELSE VDT3(LB3)=C;                                                                           |00125600
1048   3  3  |QDT3(LB3)=VDT3(LB3)*A;                                                                      |00125700
1049   3  3  |END;                                                                                        |00125800
1050   3  2  |ELSE IF ABS(PT2->PIPE_DROP+D2(HB2)-OUTFLOW_CONDITION)<.0001 THEN DO;                        |00125900
1051   3  3  |DP3=D3(LB3);                                                                                |00126000
1052   3  3  |FUN=2;  FUNPR=1;                                                                            |00126100
1054   3  3  |DO IIT=1 TO 20 WHILE(ABS(FUN/FUNPR)>.0005);                                                 |00126200
1055   3  4  |DEPTH=DP3;                                                                                  |00126300
1056   3  4  |DIAMETER=PT3->PIPE_DIAMETER;                                                                |00126400
1057   3  4  |CURPIPE=PT3;                                                                                |00126500
1058   3  4  |CALL CIRCLE;                                                                                |00126600
1059   3  4  |AA3=A;                                                                                      |00126700
1060   3  4  |BB3=B;                                                                                      |00126800
1061   3  4  |VELOCITY3=CB3+AK3*DP3;                                                                      |00126900
1062   3  4  |DP2=DP3+VELOCITY3**2/64.4-PT2->PIPE_DROP;                                                   |00127000
1063   3  4  |DIAMETER=PT2->PIPE_DIAMETER;                                                                |00127100
1064   3  4  |CURPIPE=PT2;                                                                                |00127200
1065   3  4  |CALL CIRCLE;                                                                                |00127300
1066   3  4  |AA2=A;                                                                                      |00127400
1067   3  4  |BB2=B;                                                                                      |00127500
1068   3  4  |VELOCITY2=CP2-AK2*DP2;                                                                      |00127600
1069   3  4  |IF PT3->DNORM>PT3->CRITICAL_DEPTH THEN DO;                                                  |00127700
1070   3  5  |FUN=QDT1(HB1)+VELOCITY2*AA2-VELOCITY3*AA3-(DP2-D2(HB2))                                     |00127800
              |*PT3->MANHOLE_AREA/DT+INFLOW;                                                               |00127900
1071   3  5  |FUNPR=(VELOCITY2*BB2-AK2*AA2-PT3->MANHOLE_AREA/DT)*(1+VELOCITY3*AK3                         |00128000
              |/32.2)-(VELOCITY3*BB3+AK3*AA3);                                                             |00128100
1072   3  5  |END;                                                                                        |00128200
1073   3  4  |ELSE DO;                                                                                    |00128300
1074   3  5  |DEPTH=DP3;                                                                                  |00128400
1075   3  5  |DIAMETER=PT3->PIPE_DIAMETER;                                                                |00128500
1076   3  5  |CURPIPE=PT3;                                                                                |00128600
1077   3  5  |CALL CIRCLE;                                                                                |00128700
1078   3  5  |FUN=QDT1(HB1)+VELOCITY2*AA2-C*A-(DP2-D2(HB2))*PT3->MANHOLE_AREA/DT                          |00128800
              |+INFLOW;                                                                                    |00128900
1079   3  5  |FUNPR=(VELOCITY2*BB2-AK2*AA2-PT3->MANHOLE_AREA/DT)*(1+AK3*VELOCITY3                         |00129000
              |/32.2)-C*BB3-(16.1*AA3/C)*(B**3-2*A*DIAMETER*COS(.5*THETA))/B**3;                           |00129100
1080   3  5  |END;                                                                                        |00129200
1081   3  4  |DP3=DP3-FUN/FUNPR;                                                                          |00129300
1082   3  4  |END;                                                                                        |00129400
1083   3  3  |IF IIT>20 THEN PUT SKIP LIST('*JUNCTION:  MANHOLE ITERATIONS(4) EXCEED                      |00129500
              |20.');                                                                                      |00129600
1084   3  3  |DEPTH,DDT3(LB3)=DP3;                                                                        |00129700
1085   3  3  |VDT3(LB3)=CB3+AK3*DP3;                                                                      |00129800
1086   3  3  |DIAMETER=PT3->PIPE_DIAMETER;                                                                |00129900
1087   3  3  |CURPIPE=PT3;                                                                                |00130000
1088   3  3  |CALL CIRCLE;                                                                                |00130100
1089   3  3  |QDT3(LB3)=VDT3(LB3)*A;                                                                      |00130200
1090   3  3  |DEPTH,DDT2(HB2)=DP3+VDT3(LB3)**2/64.4-PT2->PIPE_DROP;                                       |00130300
1091   3  3  |QDT2(HB2)=QDT3(LB3)-QDT1(HB1);                                                              |00130400
1092   3  3  |DIAMETER=PT2->PIPE_DIAMETER;                                                                |00130500
1093   3  3  |CURPIPE=PT2;                                                                                |00130600
1094   3  3  |CALL CIRCLE;                                                                                |00130700
1095   3  3  |VDT2(HB2)=QDT2(HB2)/A;                                                                      |00130800
1096   3  3  |END;                                                                                        |00131700
1097   3  2  |END;                                                                                        |00131800
1098   3  1  |ELSE SIGNAL ERROR;  /* WHA' HAPPENED?  WE'RE NOT SUPPOSED TO BE HERE */                    |00131900
1099   3  1  |END IT_IS_A_MANHOLE_JUNCTION;                                                               |00132000
```

```
1100   3  0  |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                    |00132100
1101   3  0  |IF BIT2 THEN BIT2=P<(LBOUND(FLOWBLOCK,1)+DIM(FLOWBLOCK,1)/4)*TI+TL;      |00132200
1102   3  0  |ELSE DO;                                                                  |00132300
1103   3  1  |DCL POST_PEAK_BASE FLOAT BIN;                                             |00132400
1104   3  1  |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                    |00132500
1105   3  1  |POST_PEAK_BASE=FLOWBLOCK(HBOUND(FLOWBLOCK,1));                            |00132600
1106   3  1  |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                    |00132700
1107   3  1  |POST_PEAK_BASE=POST_PEAK_BASE+FLOWBLOCK(HBOUND(FLOWBLOCK,1));             |00132800
1108   3  1  |IF PT3->YBLOCK_POINTER¬=NULL THEN DO;                                     |00132900
1109   3  2  |PSEUDOREGISTER=PT3->YBLOCK_POINTER;                                       |00133000
1110   3  2  |POST_PEAK_BASE=POST_PEAK_BASE+FLOWBLOCK(HBOUND(FLOWBLOCK,1));             |00133100
1111   3  2  |END;                                                                      |00133200
1112   3  1  |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                    |00133300
1113   3  1  |IF QDT3(J)<=1.1*POST_PEAK_BASE THEN BIT='0'B;                             |00133400
             |/* NOTE:  1.1  TIMES BASE ENDING FLOW FOR PREVIOUS FLOWBLOCKS. */        |00133500
1114   3  1  |END;                                                                      |00133600
             |/* BIT IS THE TERMINATION FLAG FOR THE DO-WHILE LOOP IN ROUTE. */        |00133700


1115   3  0  |JUNC_DROP:  PROC(PTR,V,D,DDT,VDT,QDT);                                    |00133800
             |/* THIS SUBROUTINE COMPUTES CRITICAL FLOW AT THE DROP STRUCTURE.     */|00133810
1116   4  0  |DCL PTR POINTER,                                                          |00133900
             |(VDT,V,D,DDT,QDT)(*) FLOAT BIN,                                           |00134000
             |I,K;                                                                      |00134100
1117   4  0  |CURPIPE=PTR;                                                              |00134200
1118   4  0  |I=HBOUND(D,1);                                                            |00134300
1119   4  0  |K=I-1;                                                                    |00134400
1120   4  0  |DIAMETER=PTR->PIPE_DIAMETER;                                              |00134500
1121   4  0  |TT=.5*DT/PTR->DELTA_X;                                                    |00134600
1122   4  0  |DEPTH=D(I);                                                               |00134700
1123   4  0  |CALL CIRCLE;                                                              |00134800
1124   4  0  |DCL DI FLOAT BIN;                                                         |00134900
1125   4  0  |DI=DM;                                                                    |00135000
1126   4  0  |DEPTH=D(K);                                                               |00135100
1127   4  0  |CALL CIRCLE;                                                              |00135200
1128   4  0  |DK=(DM+DI)*.5;                                                            |00135300
1129   4  0  |DEPTH,DDT(I)=D(I)-2*TT*(.5*(V(I)+V(K))*(D(I)-D(K))+DK*(V(I)-V(K)));       |00135400
1130   4  0  |CALL CIRCLE;                                                              |00135500
1131   4  0  |VDT(I)=C;                                                                 |00135600
1132   4  0  |QDT(I)=C*A;                                                               |00135700
1133   4  0  |END JUNC_DROP;                                                           |00135800
1134   3  0  |END JUNCTION;                                                            |00135900
1135   2  0  |END;  /* OF ((OLD ROUTE)) BEGIN BLOCK . */                               |00136000

             |/* FLOWBLOCK IS NOW FILLED WITH "FINAL" VALUES. */                       |00136100
1136   1  0  |FLOWBLOCK_FULL:                                                           |00136200
             |PRTFB=PT1->FLOWBLOCK_POINTER;                                             |00136300
1137   1  0  |PRTFB2=PT2->FLOWBLOCK_POINTER;                                            |00136400
1138   1  0  |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                    |00136500

1139   1  0  |IF GRAPHPL THEN CALL GRAPHER(TFB,TFB2,FLOWBLOCK);                         |00136600

1140   1  0  |IF INDEX(GOPARM,'PLOT')>0 THEN CALL PLOTTER;                              |00136700

             |/* NEXT:  SHRINK THE JUNCTION NODE'S FLOWBLOCK DOWN TO THE SIZE WE      |00136800
             |WANT.  NOTE THAT WE WILL ALLOW A 5% INCREASE IN FLOW OVER BASE BEFORE  |00136900
             |IT IS CONSIDERED SIGNIFICANT. */                                          |00137000
1141   1  0  |TEMP=1.05*FLOWBLOCK(SUBSCR1);                                             |00137100
1142   1  0  |DO I=SUBSCR1+1 TO SUBSCR2 WHILE(FLOWBLOCK(I)<TEMP);  END;                 |00137200
1144   1  0  |IF I>=SUBSCR2 THEN PUT EDIT('**WARNING**  FLOWBLOCK FOR NODE #',          |00137300
             |PT3->NODE_#,' HAS NO VALUES MORE THAN 5% ABOVE BASE FLOW.','IT WILL NOT|00137400
             | BE "SHRUNK".')(SKIP,A,P'ZZ,ZZ9',A,COL(14),A);                           |00137500
1145   1  0  |ELSE DO;                                                                  |00137600
             |/* FIND NEW HIGH-ORDER END. */                                           |00137700
1146   1  1  |I=I-1;                                                                    |00137800
             |/* I HAS THE FIRST SUBSCRIPT OF THE NEW FLOWBLOCK. */                     |00137900
1147   1  1  |TEMP=1.05*FLOWBLOCK(SUBSCR2);                                             |00138000
1148   1  1  |DO J=SUBSCR2-1 TO I BY -1 WHILE(FLOWBLOCK(J)<TEMP);  END;                 |00138100
1150   1  1  |IF J<=I THEN SIGNAL ERROR;                                                |00138200
1151   1  1  |IF J=SUBSCR2-1 THEN PUT EDIT('**WARNING**  FLOWBLOCK FOR NODE #',         |00138300
             |PT3->NODE_#,'''S LAST TWO VALUES ARE MORE THEN 5% APART.','THERE MAY BE|00138400
             | AN ASSOCIATED ERROR.')(SKIP,A,P'ZZ,ZZ9',A,COL(14),A);                   |00138500
1152   1  1  |ELSE DO;  /* DO THE SHRINK. */  J=J+1;                                  |00138600
1154   1  2  |K=SUBSCR1;  L=SUBSCR2;  /* SAVE VALUES TEMPORARILY. */                    |00138700
1156   1  2  |SUBSCR1=I;  SUBSCR2=J;  /* SO CAN ALLOCATE. */                           |00138800
1158   1  2  |PRTFB=PSEUDOREGISTER;                                                     |00138900
1159   1  2  |PSEUDOREGISTER=LASTALLOC;                                                 |00139000
1160   1  2  |ALLOCATE FLOWBLOCK;                                                       |00139100
```

```
                |/*                                                                                |00139200
                |PUT EDIT('*COMPUTATIONS COMPLETE--FLOWBLOCK FOR JUNCTION NODE IS BEING             |00139300
                |"SHRUNK".  THE NEW BOUNDS ARE FROM ',SUBSCR1,' TO ',SUBSCR2,'.')                  |00139400
                |(SKIP,A,P'ZZ,ZZ9',A,P'ZZ,ZZ9',A);                                                |00139500
                |*/                                                                               |00139600
1161    1   2   |LASTALLOC=PSEUDOREGISTER;                                                         |00139700
                |/* NOW: K HAS LBOUND((OLD FLOWBLOCK)TFB), L HAS HBOUND((OLD FLOWBLOCK)            |00139800
                |TFB), SUBSCR1 HAS LBOUND(NEW FLOWBLOCK), SUBSCR2 HAS HBOUND(NEW                   |00139900
                |FLOWBLOCK).  FIRST, DOUBLE CHECK AGAINST ERROR. */                                |00140000
1162    1   2   |IF K>SUBSCR1|K>SUBSCR2 THEN SIGNAL ERROR;                                         |00140100
1163    1   2   |IF L<SUBSCR2|L<SUBSCR1 THEN SIGNAL ERROR;                                         |00140200
1164    1   2   |IF K>L | SUBSCR1>SUBSCR2 THEN SIGNAL ERROR;                                       |00140300

1165    1   2   |DO I=SUBSCR1 TO SUBSCR2;                                                          |00140400
1166    1   3   |FLOWBLOCK(I)=TFB(I);                                                              |00140500
1167    1   3   |END;                                                                              |00140600
                |/* THE OLD, TOO-BIG FLOWBLOCK IS NOW MOVED OVER INTO THE NEW ONE. */             |00140700
1168    1   2   |CALL PLUCK(PT3->FLOWBLOCK_POINTER,LASTALLOC);                                     |00140800
1169    1   2   |PSEUDOREGISTER=PT3->FLOWBLOCK_POINTER;                                            |00140900
1170    1   2   |FREE FLOWBLOCK;                                                                   |00141000
1171    1   2   |IF PSEUDOREGISTER¬=LASTALLOC THEN SIGNAL ERROR;                                   |00141100
1172    1   2   |PSEUDOREGISTER,PT3->FLOWBLOCK_POINTER=LASTALLOC;  /* NOW PT TO NEW FB*/          |00141200
1173    1   2   |END;                                                                              |00141300
1174    1   1   |END;                                                                              |00141400
                |/* NOW WE HAVE THE THIRD FLOWBLOCK SHRUNK AND CHAINED.  PRINT VALUES.*/          |00141500
1175    1   0   |CALL PRTCHG;  /* PRINT THE COMPUTED HYDROGRAPH. */                                |00141600
1176    1   0   |DCL 1 QDT_ROOT CONTROLLED EXTERNAL,                                               |00141700
                |(2 ABSTIME, 2 QDTROOT, 2 DDTROOT) FLOAT BIN;                                      |00141800
1177    1   0   |IF QDT_ROOT.ABSTIME¬=-1.23456 THEN IF QDT_ROOT.QDTROOT¬=9.87654 THEN DO           |00141900
                |;                                                                                |00142000
1178    1   1   |SIGNAL ENDPAGE(SYSPRINT);                                                         |00142100
1179    1   1   |PUT EDIT('*EVALUAR DETECTS THAT LAST JUNCTION NODE HAS JUST BEEN CALCUL            |00142200
                |ATED.','APPROPRIATELY, THE DOWNSTRM SUBROUTINE SAVED THESE VALUES:',              |00142300
                |'ABS. TIME','DEPTH(ROOT NODE)','FLOW RATE(ROOT NODE)')                            |00142400
                |(A, COL(2),A,SKIP(2),COL(27),A,COL(50),A,COL(70),A);                              |00142500
1180    1   1   |DO WHILE(ALLOCATION(QDT_ROOT));                                                   |00142600
1181    1   2   |IF QDT_ROOT.ABSTIME¬=-1.23456 THEN IF QDT_ROOT.QDTROOT¬=9.87654 THEN              |00142700
                |PUT SKIP LIST(' ',QDT_ROOT.ABSTIME,QDT_ROOT.DDTROOT,QDT_ROOT.QDTROOT);           |00142800
1182    1   2   |FREE QDT_ROOT;                                                                    |00142900
1183    1   2   |END;                                                                              |00143000
1184    1   1   |PUT SKIP(2) LIST('*NOTE THAT VALUES ARE ORDERED FROM MOST RECENT TO EAR           |00143100
                |LIEST.');                                                                         |00143200
1185    1   1   |END;                                                                              |00143300

                |/* NOW, STORAGE USED FOR DYNAMICALLY ALLOCATED ARRAYS ETC., (WHICH   */          |00143310
                |/* ARE NO LONGER NEEDED) CAN BE RETURNED TO THE STORAGE POOL.        */          |00143320


                |/* FIRST WE CAN GET RID OF THE DEPTHBLOCK.                          */          |00143400
1186    1   0   |FREE DEPTHBLOCK;                                                                  |00143500


                |/* NEXT:  GET RID OF THE TWO OLD, UNNEEDED FLOWBLOCKS.              */          |00143600
                |/* AH, YES....THEY SERVED US WELL.                                  */          |00143610
1187    1   0   |PSEUDOREGISTER=PT1->FLOWBLOCK_POINTER;                                            |00143700
1188    1   0   |CALL PLUCK(PT1->FLOWBLOCK_POINTER,LASTALLOC);                                     |00143800
1189    1   0   |FREE FLOWBLOCK;  /* FREE OLD #1 FLOWBLOCK. */                                     |00143900
1190    1   0   |LASTALLOC=PSEUDOREGISTER;                                                         |00144000
1191    1   0   |PT1->FLOWBLOCK_POINTER=NULL;                                                      |00144100
1192    1   0   |PSEUDOREGISTER=PT2->FLOWBLOCK_POINTER;                                            |00144200
1193    1   0   |CALL PLUCK(PT2->FLOWBLOCK_POINTER,LASTALLOC);                                     |00144300
1194    1   0   |FREE FLOWBLOCK;  /* FREE OLD #2 FLOWBLOCK. */                                     |00144400
1195    1   0   |LASTALLOC=PSEUDOREGISTER;                                                         |00144500
1196    1   0   |PT2->FLOWBLOCK_POINTER=NULL;                                                      |00144600

                |/* NOW ONLY ONE FLOWBLOCK REMAINS--THE ONE FOR NODE #3, THE JUNCTION   |00144700
                |NODE.  ITS FLOWBLOCK HAS BEEN CALCULATED AND PRINTED.  THE ONLY THING  |00144800
                |REMAINING TO DO IS TO GET RID OF ALL OF THE WORK ARRAYS. */            |00144900
1197    1   0   |FREE DDT1,DDT2,DDT3,D1,D2,D3,QDT1,QDT2,QDT3,Q1,Q2,Q3,VDT1,VDT2,VDT3,   |00145000
                |V1,V2,V3;                                                              |00145100
                |/* BY NOW WE SHOULD HOPEFULLY HAVE FREED ALL STORAGE ALLOCATED THAT IS |00145200
                |NOT SUPPOSED TO STICK AROUND. */                                       |00145300

1198    1   0   |(NOSUBSCRIPTRANGE):                                                               |00145400
                |DNORMAL:  PROC(NPT,D,Q,V,QDT);                                                    |00145500
                |/* COMPUTES NORMAL DEPTH CORRESPONDING TO INITIAL BASE FLOW IN PIPE  */          |00145510
1199    2   0   |DCL (NPT,TEMPPTR) POINTER,                                                        |00145600
                |TVAL FLOAT BIN,                                                                  |00145700
                |(D(*),V(*),Q(*),QDT(*)) FLOAT BIN,                                               |00145800
                |DTT FLOAT BIN;                                                                   |00145900
1200    2   0   |PSEUDOREGISTER=NPT->FLOWBLOCK_POINTER;                                            |00146000
1201    2   0   |QB=FLOWBLOCK(LBOUND(FLOWBLOCK,1));                                                |00146100
1202    2   0   |DIAMETER=NPT->PIPE_DIAMETER;                                                      |00146200
```

```
1203   2  0  |CURPIPE=NPT;                                                                    |00146300
1204   2  0  |FN=NPT->PIPE_ROUGHNESS;                                                         |00146400
1205   2  0  |DEPTH=.3*DIAMETER;                                                              |00146500
1206   2  0  |DN=DEPTH+4092;                                                                  |00146600
1207   2  0  |DO IIT=1 TO 20 WHILE(ABS(DN-DEPTH)>.0005);                                      |00146700
1208   2  1  |CALL CIRCLE;                                                                    |00146800
1209   2  1  |VV=QB/A;                                                                        |00146900
1210   2  1  |TVAL=LOG10(2*R/EN);                                                             |00147000
1211   2  1  |REY=VV*R/1.2E-5;                                                                |00147100
1212   2  1  |RSTR=.633*(TVAL+.87)**8;                                                        |00147200
1213   2  1  |IF REY>=RSTR THEN F=1/(2*TVAL+1.74)**2;                                         |00147300
1214   2  1  |ELSE F=.223/SQRT(SQRT(REY));                                                    |00147400
1215   2  1  |DN=DEPTH;                                                                       |00147500
1216   2  1  |DEPTH=DEPTH-(WP-(F*QB*QB) /(257.6*NPT->PIPE_SLOPE*R*R*A))/((3*B)/R-             |00147600
              |(2*DIAMETER/B));  /* SOME OF THIS COULD BE TAKEN OUT OF DO-WHILE LOOP*/       |00147700
                                                                                            |00147800
1217   2  1  |END;                                                                            |00147900
1218   2  0  |IF IIT>=20 THEN PUT SKIP LIST('*DNORMAL:  ITERATIONS EXCEED 20.');              |00148000
              |/* EN IS ROUGHNESS, A,R,B SET BY CIRCL, FNU IS FROM EVALUBR. */               |00148100
1219   2  0  |CALL CIRCLE;                                                                    |00148200
1220   2  0  |VV=QB/A;                                                                        |00148300
1221   2  0  |DTT=NPT->DELTA_X/ABS(VV+C);  /* C COMES FROM CIRCLE. */                         |00148400
1222   2  0  |IF DTT<DT THEN DT=DTT;                                                          |00148500
1223   2  0  |D=DEPTH;  /* INITIALIZES THE ARRAY. */                                          |00148600
1224   2  0  |V=VV;  /* DITTO. */                                                             |00148700
1225   2  0  |Q=QB;  /* DITTO AGAIN. */                                                       |00148800
1226   2  0  |QDT=QB;                                                                         |00148900
1227   2  0  |END DNORMAL;
```

```
1228   1  0  |(NOSUBSCRIPTRANGE):                                                             |00149700
              |DCRIT:  PROC(NODE_PTR) RETURNS(FLOAT BIN);                                     |00149800
              |/* COMPUTES CRITICAL FLOW DEPTH CORRESPONDING TO INITIAL BASE FLOW.  */       |00149810
1229   2  0  |DCL NODE_PTR POINTER, DCR FLOAT BIN;                                            |00149900
1230   2  0  |PSEUDOREGISTER=NODE_PTR->FLOWBLOCK_POINTER;                                     |00150000
1231   2  0  |DIAMETER=NODE_PTR->PIPE_DIAMETER;                                               |00150100
1232   2  0  |CURPIPE=NODE_PTR;                                                               |00150110
1233   2  0  |DEPTH=.3*DIAMETER;                                                              |00150200
1234   2  0  |DCR=5000000;                                                                    |00150300
1235   2  0  |DCL I FIXED BIN(15);                                                            |00150400
1236   2  0  |DO I=1 TO 20 WHILE(ABS(DCR-DEPTH)>.0005);                                       |00150500
1237   2  1  |IF I>1 THEN DEPTH=DCR;                                                          |00150600
1238   2  1  |CALL CIRCLE;                                                                    |00150700
1239   2  1  |DCR=DEPTH-(B*(A**3)-((B*FLOWBLOCK(LBOUND(FLOWBLOCK,1)))**2)/32.2)              |00150800
              |/(3.*((B*A)**2)-(2*(A**3)*COS(THETA*.5))/SIN(THETA*.5));                       |00150900
1240   2  1  |END;                                                                            |00151000
1241   2  0  |PUT SKIP;                                                                       |00151100
1242   2  0  |IF I<19 THEN RETURN(DCR);                                                       |00151200
1243   2  0  |PUT SKIP LIST('*DCRIT ITERATIONS EXCEED 20.');  SIGNAL ERROR;                   |00151300
1245   2  0  |END DCRIT;
```

```
1246   1  0  |INCOND:  PROC(PTR, D, V);                                                       |00151500
              |/* COMPUTES INITIAL STEADY NONUNIFORM FLOW AND BACKWATER PROFILES    */       |00151510
1247   2  0  |DCL PTR POINTER,                                                                |00151600
              |(D, V)(*) FLOAT BIN,                                                           |00151700
              |(NWV2,DY, QQB, F) FLOAT BIN,                                                   |00151800
              |(IT, ICOUNT INIT(1)) FIXED BIN,                                                |00151900
              |1 PR, (2 D, 2 X, 2 SF, 2 E, 2 V) FLOAT BIN,                                    |00152000
              |1 NW LIKE PR;                                                                  |00152100
1248   2  0  |CURPIPE=PTR;                                                                    |00152110
1249   2  0  |PSEUDOREGISTER=PTR->FLOWBLOCK_POINTER;                                          |00152200
1250   2  0  |QQB=FLOWBLOCK(LBOUND(FLOWBLOCK,1));                                            |00152300
1251   2  0  |PR.X=0;                                                                         |00152400
1252   2  0  |DIAMETER=PTR->PIPE_DIAMETER;                                                    |00152500
1253   2  0  |IF PTR->DNORM>PTR->CRITICAL_DEPTH THEN DO;                                      |00152600
1254   2  1  |DEPTH,PR.D,D(HBOUND(D,1))=CONTROL;                                              |00152700
1255   2  1  |DY=(D(LBOUND(D,1))-D(HBOUND(D,1))) / 20;                                        |00153200
1256   2  1  |CALL CIRCLE;                                                                    |00153300
1257   2  1  |V(HBOUND(V,1)),  PR.V=QQB/A;                                                    |00153400
1258   2  1  |END;                                                                            |00153500
1259   2  0  |ELSE DO;                                                                        |00153600
1260   2  1  |DEPTH, PR.D, D(LBOUND(D,1))=CONTROL;                                            |00153700
1261   2  1  |DY=(D(HBOUND(D,1))-D(LBOUND(D,1)))/20;                                          |00153800
1262   2  1  |CALL CIRCLE;                                                                    |00153900
1263   2  1  |V(LBOUND(V,1)),PR.V=QQB/A;                                                      |00154000
1264   2  1  |END;                                                                            |00154100
1265   2  0  |TVAL=LOG10(2*R/PTR->PIPE_ROUGHNESS);                                           |00154200
1266   2  0  |REY=ABS(PR.V*R/1.2E-5);                                                         |00154210
1267   2  0  |RSTR=.633*(TVAL+.87)**8;                                                        |00154220
1268   2  0  |IF REY<=RSTR THEN F=.223/REY**.25;                                              |00154230
1269   2  0  |ELSE F=1/(2*TVAL+1.74)**2;                                                      |00154240
1270   2  0  |PR.SF=F*PR.V*PR.V/(257.6*R);                                                    |00154300
```

```
1271   2  0  |PR.E=PR.D+PR.V*PR.V/64.4;                                                    |00154400
1272   2  0  |DO IT=1 TO 20 WHILE(ICOUNT<=DIM(D,1)-1);                                    |00154500
1273   2  1  |DEPTH, NW.D=PR.D+DY;                                                         |00154600
1274   2  1  |IF ABS(NW.D-D(LBOUND(D,1)))<.01 THEN RETURN;                                 |00154700
1275   2  1  |CALL CIRCLE;                                                                 |00155000
1276   2  1  |NW.V=QQB/A;                                                                  |00155100
1277   2  1  |NWV2=NW.V**2;                                                                |00155200
1278   2  1  |TVAL=LOG10(2*R/PTR->PIPE_ROUGHNESS);                                         |00155300
1279   2  1  |REY=ABS(NW.V*R/1.2E-5);                                                      |00155310
1280   2  1  |RSTR=.633*(TVAL+.87)**8;                                                     |00155320
1281   2  1  |IF REY<=RSTR THEN F=.223/REY**.25;                                           |00155330
1282   2  1  |ELSE F=1/(2*TVAL+1.74)**2;                                                   |00155340
1283   2  1  |NW.SF=F*NWV2/(257.6*R);                                                      |00155400
1284   2  1  |NW.E=NW.D+NWV2/64.4;                                                         |00155500
1285   2  1  |NW.X=PR.X+SIGN(PTR->DNORM-PTR->CRITICAL_DEPTH)*(PR.E-NW.E)/(                 |00155600
              |PTR->PIPE_SLOPE-.5*(NW.SF+PR.SF));                                          |00155700
1286   2  1  |IF NW.X>ICOUNT*PTR->DELTA_X THEN DO;                                         |00155800
1287   2  2  |IF PTR->DNORM>PTR->CRITICAL_DEPTH THEN DO;                                   |00155900

1288   2  3  |DEPTH, D(HBOUND(D,1)-ICOUNT)=PR.D+(NW.D-PR.D)*(ICOUNT*PTR->DELTA_X          |00156000
              |-PR.X) / (NW.X-PR.X);                                                      |00156100
1289   2  3  |CALL CIRCLE;                                                                 |00156200
1290   2  3  |V(HBOUND(V,1)-ICOUNT)=QQB/A;                                                 |00156300
1291   2  3  |END;                                                                         |00156400
1292   2  2  |ELSE DO;                                                                     |00156500
1293   2  3  |DEPTH,D(LBOUND(D,1)+ICOUNT)=PR.D+(NW.D-PR.D)*(ICOUNT*PTR->DELTA_X           |00156600
              |-PR.X)/(NW.X-PR.X);                                                         |00156700
1294   2  3  |CALL CIRCLE;                                                                 |00156800
1295   2  3  |V(LBOUND(V,1)+ICOUNT)=QQB/A;                                                 |00156900
1296   2  3  |END;                                                                         |00157000
1297   2  2  |ICOUNT=ICOUNT+1;                                                             |00157100
1298   2  2  |END;                                                                         |00157200
1299   2  1  |PR=NW;                                                                       |00157300
1300   2  1  |END /* OF DO-WHILE LOOP IN INCOND */ ;                                       |00157400
1301   2  0  |IF IT>=20 THEN DO;                                                           |00157500
1302   2  1  |PUT SKIP(3) LIST('*INCOND:  LOOP COUNT EXCEEDED.');                          |00157600
1303   2  1  |END;                                                                         |00157700
1304   2  0  |END INCOND;                                                                  |00157800


1305   1  0  |(NOSUBSCRIPTRANGE):                                                          |00159100
              |CIRCLE:  PROC REORDER;                                                      |00159200
              |/* THIS SUBROUTINE COMPUTES GEOMETRICAL PARAMETERS OF CIRCULAR PIPES */    |00159203

1306   2  0  |ON ERROR BEGIN;                                                              |00159205
1307   3  0  |DCL I FIXED BIN, CURPIPE POINTER EXTERNAL;                                   |00159210
1308   3  0  |DCL PDIAMS(0:35) FLOAT BIN STATIC EXTERNAL;                                  |00159213
1309   3  0  |DO I=1 TO HBOUND(PDIAMS,1) WHILE(PDIAMS(I)<CURPIPE->PIPE_DIAMETER);         |00159215
1310   3  1  |END;                                                                         |00159220
1311   3  0  |PUT EDIT('* FOR THE PIPE DOWNSTREAM FROM NODE #',CURPIPE->NODE_#,           |00159225
              |'--THE CHOSEN DIAMETER,',CURPIPE->PIPE_DIAMETER,                            |00159230
              |' FEET IS TOO SMALL.  THE DIAMETER IS BEING','INCREASED TO',                |00159235
              |PDIAMS(I+1),' FEET, SO THAT SIMULATION ON THIS TRIPLET CAN BE RESTARTED     |00159240
              |.')                                                                         |00159245
              |(SKIP(2),A,P'ZZ,ZZ9',A,P'Z9V.99',A,SKIP,X(5),A,P'Z9V.99',A);               |00159250
1312   3  0  |CURPIPE->PIPE_DIAMETER=PDIAMS(I+1);                                          |00159255
1313   3  0  |CURPIPE->CRASHED='1'B;                                                       |00159258
1314   3  0  |GOTO RESTART_POINT;                                                          |00159260
1315   3  0  |END;                                                                         |00159265
1316   2  0  |IF DIAMETER/2=DEPTH THEN THETA=3.14159;                                      |00159900
1317   2  0  |ELSE DO;                                                                     |00160000
1318   2  1  |IF DIAMETER/2<DEPTH THEN                                                     |00160100
              |THETA=6.218318-2*  ATAN((SQRT(DIAMETER*DEPTH-DEPTH*DEPTH))                 |00160200
              |/(DEPTH-DIAMETER/2));                                                       |00160300
1319   2  1  |ELSE DO;                                                                     |00160400
1320   2  2  |THETA=2*  ATAN((SQRT(DIAMETER*DEPTH-DEPTH*DEPTH))/(DIAMETER/2-DEPTH));     |00160500
1321   2  2  |IF THETA<0 THEN THETA=THETA+6.28318;                                         |00160600
1322   2  2  |END;                                                                         |00160700
1323   2  1  |END;                                                                         |00160800
1324   2  0  |A=.125*(THETA-SIN(THETA))*DIAMETER*DIAMETER;                                 |00160900
1325   2  0  |WP=DIAMETER*THETA/2;                                                         |00161000
1326   2  0  |B=DIAMETER*SIN(THETA/2);                                                     |00161100
1327   2  0  |R=A/WP;                                                                      |00161300
1328   2  0  |DM=A/B;                                                                      |00161400
1329   2  0  |AK=SQRT(32.2/DM);                                                            |00161500
1330   2  0  |C=AK*DM;                                                                     |00161600
1331   2  0  |END CIRCLE;                                                                  |00161700
1332   1  0  |END EVALUBR;                                                                 |00161800
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	#_ELEMENTS	AUTOMATIC ALIGNED BINARY FIXED (15,0) 3,4,4,4,4,4,4,6,7,7,7,7,7,9,10,10,10,10,10,10
********	A	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 348,364,365,376,376,376,384,400,423,446,465,496,502,513,517,525,529,733,739, 745,753,763,769,775,787,793,801,811,817,835,836,847,847,847,853,864,870,878, 888,894,918,925,932,945,954,960,966,978,985,998,1007,1013,1033,1034,1037, 1038,1038,1038,1048,1059,1066,1078,1079,1089,1095 564,570,582,615,621,633,672,700,1132,1209,1216,1220,1239,1239,1239),1257, 1263,1276,1290,1295,1324,1327,1328
732	AA1	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 733,748,749,752,753,787,796,797,800,801,925,936,937,944,945,985,989,990,997, 998
732	AA2	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 739,748,749,752,753,864,873,874,877,878,932,936,937,944,945,1066,1070,1071, 1078,1079
732	AA3	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 745,748,749,752,753,793,796,797,800,801,870,873,874,877,878,918,936,937,944, 945,978,989,990,997,998,1059,1070,1071,1079
********	ABS	BUILTIN 344,361,373,385,419,432,436,442,456,460,466,486,727,782,832,844,859,909,909, 913,973,1015,1017,1026,1050,1054 578,591,595,629,642,646,668,681,692,1207,1221,1236,1266,1274,1279
1176	ABSTIME	/* IN QDT_ROOT EXTERNAL */ CONTROLLED ALIGNED BINARY /* SINGLE */ FLOAT (21) 37,1177,1181,1181 539
********	ADDR	BUILTIN 31,32,33,34,158,159 15
********	AK	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 356,357,429,452,588,596,639,647,678,682,1329,1330
405	AKR	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 429,461,461,462
405	AKS	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 452,461,461
320	AK1	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 357,364,365
544	AK1	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 748,749,749,752,753,753,759,796,797,797,800,801,801,807,927,937,945,956,987, 990,998 588
544	AK2	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 748,749,749,752,753,753,765,873,874,874,877,878,878,884,934,937,945,962, 1068,1071,1079 639
544	AK3	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 748,749,749,796,797,797,835,836,873,874,874,920,937,937,945,950,980,990,990, 998,1003,1027,1034,1034,1046,1061,1071,1071,1079,1085 678
********	ALLOCATION	BUILTIN 1180
********	ATAN	BUILTIN 1318,1320
254	ATIME	(2) /* IN Q_AND_D_VALUES */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 258,276,277,277,279,280,289,289,289,296,296,299

```
********  B                    AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                               348,348,348,365,376,376,376,423,423,446,446,734,740,746,753,753,788,794,801,
                               801,836,847,847,847,865,871,878,878,919,926,933,945,945,945,979,986,998,998,
                               1034,1038,1038,1038,1038,1038,1060,1067,1079,1079
                               582,582,582,633,633,633,633,633,672,672,672,1216,1216,1239,1239,1239,1326,
                               1328

732       BB1                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               734,749,753,788,797,801,926,937,945,986,990,998

732       BB2                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               740,749,753,865,874,878,933,937,945,1067,1071,1079

732       BB3                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               746,749,753,794,797,801,871,874,878,919,937,945,979,990,998,1060,1071,1079

255       BIT                STATIC EXTERNAL UNALIGNED BIT (1)
                               256,312

                               285,329,1113

255       BIT2               AUTOMATIC UNALIGNED INITIAL BIT (1)
                               253,1101,1101

********  C                    AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                               347,348,354,385,396,397,422,423,430,445,446,453,454,466,495,496,752,753,753,
                               800,801,801,877,878,878,944,945,945,997,998,998,1037,1037,1038,1038,1047,
                               1078,1079,1079
                               569,570,581,582,589,620,621,632,633,640,671,672,680,697,698,1131,1132,1221,
                               1330

********  CB                   AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                               356,364

544       CB3                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               748,749,796,797,835,873,874,920,950,980,1003,1027,1046,1061,1085
                               682

544       CF1                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               748,749,752,753,759,796,797,800,801,807,927,956,987
                               596

544       CF2                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               748,749,752,753,765,873,874,877,878,884,934,962,1068
                               647

76        CHOP               AUTOMATIC ALIGNED BINARY FIXED (15,0)
                               80,81,83,84,85,85

1305      CIRCLE             ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                               346,352,363,375,382,389,399,421,428,444,451,464,480,483,494,501,524,731,738,
                               744,762,768,774,786,792,810,816,834,846,852,863,869,887,893,917,924,931,943,
                               953,959,965,977,984,996,1006,1012,1031,1045,1058,1065,1077,1088,1094
                               563,568,580,587,604,614,619,631,638,655,670,677,690,699,1123,1127,1130,1208,
                               1219,1238,1256,1262,1275,1289,1294

552       COEFFICIENT        ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                               722,907

208       CONTROL            AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               210,214,218,225,226,230,235,236,240,1254,1254,1254,1260,1260,1260

********  COS                BUILTIN
                               376,753,801,847,878,945,998,1038,1038,1079,1239

2         CRASHED            /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                               68,1313

2         CRITICAL_DEPTH     /* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               202,203,204,205,209,210,218,224,225,225,230,234,235,235,240,273,339,387,411,
                               411,437,453,747,778,795,822,824,829,872,935,969,988,1016,1018,1032,1046,
                               1069
                               555,566,567,606,617,618,663,1253,1285,1287

1307      CURPIPE            STATIC EXTERNAL ALIGNED POINTER
                               1309,1311,1311,1312,1313

2         CURPIPE            STATIC EXTERNAL ALIGNED POINTER
                               321,402,408,472,541,730,737,743,761,767,773,785,791,809,815,828,862,868,886,
                               892,916,923,930,942,952,958,964,976,983,995,1005,1011,1029,1044,1057,1064,
                               1076,1087,1093
                               560,603,611,654,661,1117,1203,1232,1248

319       D                  (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                               336,340,340,341,347,358,371,388,393,393,395,395,395,396
```

```
1247        D                   /* IN NW */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1273,1274,1284,1288,1293

405         D                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                415,415,416,422,437,437,439,445

1116        D                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1118,1122,1126,1129,1129,1129

1247        D                   /* IN PR */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1254,1260,1271,1273,1288,1288,1293,1293

1247        D                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1254,1254,1255,1255,1255,1255,1260,1260,1261,1261,1261,1261,1272,1274,1274,
                                1288,1288,1293,1293

1199        D                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1223

301         DATACOUNT           AUTOMATIC ALIGNED INITIAL BINARY FIXED (31,0)
                                253,302,302,303

1229        DCR                 AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1234,1236,1237,1239,1242

1228        DCRIT               ENTRY RETURNS(BINARY /* SINGLE */ FLOAT (21))
                                202,204,205

405         DDT                 (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                461,462,463

1116        DDT                 (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1129

319         DDT                 (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                380,396,398

1176        DDTROOT             /* IN QDT_ROOT EXTERNAL */ CONTROLLED ALIGNED BINARY /* SINGLE */ FLOAT
                                (21)
                                1181
                                538

2           DDT1                (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                4,4,5,245,1197
                                266,268,758,759,806,807,822,955,956,1008,1016

2           DDT2                (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                7,7,8,247,1197
                                267,269,764,765,778,824,883,884,961,962,969,1018,1090

2           DDT3                (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                10,10,11,222,1197
                                272,275,275,491,493,500,506,523,538,771,812,851,890,949,1002,1019,1043,1084
                                697

2           DEBUG               STATIC EXTERNAL UNALIGNED CHARACTER (20) VARYING
                                248
                                106,113,901,908
                                553

2           DELTA_X             /* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                3,6,9,338,385,397,406,466,484,573,624,662,698,1121,1221,1286,1288,1293

2           DEPTH               AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                345,351,362,371,373,374,376,380,388,398,420,427,443,450,463,479,482,493,500,
                                728,735,741,758,764,771,783,789,806,812,833,842,844,845,847,851,860,866,883,
                                890,914,940,949,955,961,974,993,1002,1008,1030,1043,1055,1074,1084,1090
                                562,567,579,586,601,613,618,630,637,652,669,676,689,697,1122,1126,1129,1205,
                                1206,1207,1215,1216,1216,1223,1233,1236,1237,1239,1254,1260,1273,1288,1293,
                                1316,1318,1318,1318,1318,1318,1320,1320,1320,1320

2           DEPTHBLOCK          (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                151,151,152,1186
                                292

********    DESIGN              AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                599,650

1124        DI                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1125,1128

2           DIAMETER            AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                326,348,348,407,409,473,729,736,742,753,760,766,772,784,790,801,808,814,827,
                                861,867,878,885,891,915,922,929,941,945,951,957,963,975,982,994,998,1004,
                                1010,1028,1038,1038,1056,1063,1075,1079,1086,1092
                                559,582,582,602,610,633,633,653,660,672,672,1120,1202,1205,1216,1231,1233,
                                1252,1316,1318,1318,1318,1320,1320,1324,1324,1325,1326
```

405	DIAMETER2	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 409,423,446
57	DIAMSET	ENTRY RETURNS(BINARY /* SINGLE */ FLOAT (21)) 91,92
********	DIM	BUILTIN 1101,1272
1021	DISCRG	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 1022,1033,1037
********	DK	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 1128,1129
********	DM	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 396,396,481,491,697,697,1125,1128,1328,1329,1330
********	DMI	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 481,491
********	DN	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 1206,1207,1215
2	DNORM	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21) 196,198,200,209,224,234,273,339,387,411,411,437,453,747,778,795,822,824,829, 872,935,969,988,1016,1018,1032,1046,1069 555,606,663,1253,1285,1287
1198	DNORMAL	ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 195,197,199
470	DNSTRM	ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 273
2	DOWNSTREAM_PIPE_INFO	/* IN NODE */ BASED /* STRUCTURE */
319	DP	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 358,362,364,366,366,372,373,374,376,380,380
********	DP	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 724,728,735,741,748,748,748,749,749,749,752,752,753,753,755,755,758,764,771, 771,779,783,789,796,796,797,797,800,801,803,803,806,812,812,826,833,835,837, 837,843,844,845,847,851,851,856,860,866,873,873,874,874,877,878,880,880,883, 890,890
********	DP1	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 921,927,936,944,981,987,989,997
********	DP2	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 928,934,1062,1068,1070,1078
********	DP3	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 910,914,920,921,928,940,947,947,949,949,950,955,961,970,974,980,981,993, 1000,1000,1002,1002,1003,1008,1019,1027,1030,1033,1037,1040,1040,1043,1043, 1046,1051,1055,1061,1062,1074,1081,1081,1084,1084,1085,1090
405	DR	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 416,420,422,424,424,427,461,462
********	DR	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 575,579,581,583,583,586,596,626,630,632,634,634,637,647
********	DS	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 665,669,671,673,673,676,682
319	DS	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 341,345,347,349,349,351,356
405	DS	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 439,443,445,447,447,450,461
********	DT	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 194,206,264,300,338,356,406,461,462,484,499,936,937,944,945,989,990,997,998, 1023,1070,1071,1078,1079 573,596,624,647,662,682,1121,1222,1222
1199	DTT	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 1221,1222,1222
14	DUMBPTR	AUTOMATIC ALIGNED POINTER 15,16
********	DV	AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6) 392,396,397,685,697,698

```
254         DVAL                        (2) /* IN Q_AND_D_VALUES */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT
                                        (21)
                                        260,275,292,292,292,298,298,299

1247        DY                          AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

                                        1255,1261,1273

********    DY                          AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        393,396,397,686,697,698

2           D1                          (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        4,4,5,195,196,196,227,231,245,251,1197
                                        266,268,714,723,821,822,903,909,936,944,989,997,1015,1016
                                        553,557,561,562,566,567,572,574,574,575,581,600,600,601,601

2           D2                          (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        7,7,8,197,198,198,237,241,247,251,1197
                                        267,269,715,723,777,778,823,824,855,903,909,969,1017,1018,1050,1070,1078
                                        553,608,612,613,617,618,623,625,625,626,632,651,651,652,652

********    D2V                         AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        394,396,397,687,697,698

********    D2Y                         AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        395,396,397,688,697,698

2           D3                          (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        10,10,11,199,200,200,211,215,219,222,223,223,251,1197
                                        260,272,479,482,491,491,491,499,499,716,722,723,723,724,777,779,821,823,826,
                                        842,855,856,903,906,910,970,1019,1020,1051
                                        553,599,599,600,600,650,650,651,651,658,664,664,665,671,686,686,688,688,688,
                                        689,697

1247        E                           /* IN NW */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        1284,1285

1247        E                           /* IN PR */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        1271,1285

********    EN                          AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        1204,1210

116         ENDBLOCK                    /* STATEMENT LABEL CONSTANT */
                                        110

468         ENDLOOP                     /* STATEMENT LABEL CONSTANT */
                                        414

1           EVALUBR                     EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

264         EX_DO_WHILE_LOOP            /* STATEMENT LABEL CONSTANT */
                                        312

********    E3                          AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)

                                        1020,1033,1037

1247        F                           AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        1268,1269,1270,1281,1282,1283

405         F                           AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        434,435,436,458,459,460

********    F                           AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        353,355,390,391,593,594,595,644,645,646,679,681,694,695,696,1213,1214,1216

471         F                           AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        488,489,490,509

2           FDE                         /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        506,523

2           FDI                         /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                        507,512,513,528

278         FIND_TEMP_FROM_FLOWINDEX    /* STATEMENT LABEL CONSTANT */
                                        294

75          FINDIAM                     ENTRY RETURNS(BINARY /* SINGLE */ FLOAT (21))
                                        112
                                        65

14          FIXBIN                      BASED (FIXBINPTR) ALIGNED BINARY FIXED (31,0)
```

```
********  FIXBINPTR                    AUTOMATIC ALIGNED POINTER
                                       15

76        FLOW_RATE_FOR_DIAMETER       AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       82,83

2         FLOWBLOCK                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       42,42,46,47,50,51,51,53,118,119,120,122,123,124,149,149,150,155,156,156,165,
                                       174,177,177,182,191,1139,1141,1142,1147,1148,1160,1160,1166,1170,1189,1194
                                       101,101,102,102,102,104,104,105,105,105
                                       62,62,63,141,141,142,143,258,259,260,261,262,290,314,315,316
                                       284,332,332,332,333,333,414,414,478,478,547,547,549,549,1101,1101,1105,1105,
                                       1107,1107,1110,1110
                                       708,708,708,709,709,709,710,710,710,1201,1201,1239,1239,1250,1250

274       FLOWBLOCK_BOUND_EXCEEDED     /* STATEMENT LABEL CONSTANT */
                                       330

1136      FLOWBLOCK_FULL               /* STATEMENT LABEL CONSTANT */
                                       286

2         FLOWBLOCK_POINTER            /* IN NODE */ BASED ALIGNED POINTER
                                       39,40,41,44,48,49,52,117,121,153,154,160,161,173,176,179,190,193,1136,1137,
                                       1138,1168,1169,1172,1187,1188,1191,1192,1193,1196
                                       98,99
                                       60,257,281,313,325,410,477,546,548,1100,1104,1106,1112,1200,1230,1249

255       FLOWINDEX                    AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                       261,278,290,292,293,293,314,315,315,315,316
                                       284

405       FPTR                         AUTOMATIC ALIGNED POINTER

2         FT                           /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       505

405       FUN                          AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       417,419,422,424,440,442,445,447

319       FUN                          AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       359,361,364,366

544       FUN                          AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       725,727,748,752,755,780,782,796,800,803,830,832,835,837,857,859,873,877,880,
                                       911,913,936,944,947,971,973,989,997,1000,1024,1026,1033,1037,1040,1052,1054,
                                       1070,1078,1081
                                       576,578,581,583,627,629,632,634,666,668,671,673

319       FUNC                         AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       342,344,347,349

319       FUNCPR                       AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       343,344,348,349

2         FUNCPTR                      /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                       POINTER
                                       213,478,504,509

2         FUNCTION_INFO                /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */

2         FUNCVARA                     /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                       BINARY FIXED (15,0)
                                       511,515,522,527

2         FUNCVARR                     /* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED
                                       BINARY FIXED (15,0)
                                       510,521,521

319       FUNPR                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       360,361,365,366

405       FUNPR                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       418,419,423,424,441,442,446,447

544       FUNPR                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       726,727,749,753,755,781,782,797,801,803,831,832,836,837,858,859,874,878,880,
                                       912,913,937,945,947,972,973,990,998,1000,1025,1026,1034,1038,1040,1053,1054,
                                       1071,1079,1081
                                       577,578,582,583,628,629,633,634,667,668,672,673

2         FV                           /* IN OUTVARS EXTERNAL */ STATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       508,516,517

30        GETFBPR                      EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                       32
```

30	GETTPR	EXTERNAL ENTRY(ALIGNED POINTER) RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 33,158
30	GETTGPR	EXTERNAL ENTRY(ALIGNED POINTER) RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 34,159
30	GET1BPR	EXTERNAL ENTRY(ALIGNED POINTER) RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 31
2	GOPARM	STATIC EXTERNAL UNALIGNED CHARACTER (20) VARYING 19,20,20,1140
2	GRAPHER	EXTERNAL ENTRY((*) ALIGNED BINARY /* SINGLE */ FLOAT (21),(*) ALIGNED BINARY /* SINGLE */ FLOAT (21),(*) ALIGNED BINARY /* SINGLE */ FLOAT (21)) RETURNS(DECIMAL /* SINGLE */ FLOAT (6)) 1139
2	GRAPHFL	STATIC EXTERNAL UNALIGNED BIT (1) 1139
56	GUESS	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 112,113,114,115
2	HBFB1	AUTOMATIC ALIGNED BINARY FIXED (31,0) 119,126,147,164,170
2	HBFB2	AUTOMATIC ALIGNED BINARY FIXED (31,0) 123,127,147,181,187
********	HBOUND	BUILTIN 42,47,119,123,196,198 101,101,101,102,102,102,104,104,105,105 62,78,143,315,411,414,474,547,549,714,715,1105,1107,1110 556,557,600,601,607,608,651,652,709,709,1118,1254,1255,1257,1261,1288,1290, 1309
544	HB1	AUTOMATIC ALIGNED BINARY FIXED (31,0) 714,723,758,759,759,763,763,770,806,807,807,811,811,813,821,825,873,877,889, 903,903,909,936,944,955,956,956,960,960,989,997,1008,1009,1013,1013,1015, 1022,1070,1078,1091 553,553
544	HB2	AUTOMATIC ALIGNED BINARY FIXED (31,0) 715,718,723,764,765,765,769,769,770,777,796,800,813,823,825,855,883,884,884, 888,888,889,903,903,909,961,962,962,966,966,989,997,1009,1017,1022,1050, 1070,1078,1090,1091,1095,1095 553,553
58	HOLD	AUTOMATIC ALIGNED POINTER 59,72
1116	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 1118,1119,1122,1129,1129,1129,1129,1129,1131,1132
1307	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 1309,1309,1309,1311,1312
405	I	AUTOMATIC ALIGNED BINARY FIXED (31,0) 411,411,412,413,415,415,416,422,422,430,430,430,437,437,439,445,445,453,453, 453,454,454,454,461,462,462,463,465,465,466
319	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 335,337,340,340,340,340,341,347,347,354,354,354,354,354,383,384,385
1235	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 1236,1236,1237,1242
76	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 79,79
471	I	AUTOMATIC ALIGNED BINARY FIXED (31,0) 475,478,479,491,491,491,499,499,537,538
2	I	AUTOMATIC ALIGNED BINARY FIXED (31,0) 46,51,126,128,130,164,164,165,165,167,167,168,168,170,170,171,171,181,181, 182,182,184,184,185,185,187,187,188,188,1142,1142,1142,1144,1146,1146,1148, 1150,1156,1165,1165,1166,1166 101,101,102,102,102,104,104,105,105,106 62,62,63,304,304,305,306,556,558,561,562,564,564,566,567,569,570,573,574, 574,574,574,575,581,581,589,589,589,589,589,607,609,612,613,615,615,617,618, 620,621,623,625,625,625,625,626,632,632,640,640,640,640,640,657,659,664,664, 664,664,665,671,671,680,680,680,680,680

```
1247        ICOUNT                      AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                        1246,1272,1286,1288,1288,1290,1293,1293,1295,1297,1297

********    IIT                         AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        361,361,368,727,727,757,782,782,805,832,832,839,844,844,845,849,859,859,882,
                                        913,913,973,973,1026,1026,1042,1054,1054,1083,1207,1207,1218

2           IMAGINARY                   /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                        42,197,203,233
                                        92,96,267,269,718
                                        606

1246        INCOND                      ENTRY RETURNS(BINARY FIXED (15,0))
                                        211,215,219,227,231,237,241

********    INDEX                       BUILTIN
                                        19,20,248,1140
                                        106,113,901,908
                                        553

717         INFLOW                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                        719,720,748,752,770,796,800,813,825,873,877,936,944,989,997,1022,1070,1078

2           INPUT_STATION               /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                        162,180

404         INTER                       ENTRY RETURNS(BINARY FIXED (15,0))
                                        268,269,272

1247        IT                          AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        1272,1272,1301

********    IT                          AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        344,344,373,373,374,378,578,578,585,629,629,636,668,668,675

900         IT_IS_A_MANHOLE_JUNCTION    /* STATEMENT LABEL CONSTANT */

721         IT_IS_A_Y_JUNCTION          /* STATEMENT LABEL CONSTANT */

783         ITERATION                   /* STATEMENT LABEL CONSTANT */

319         J                           AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        336,337,358,371,380,381,383,388,391,392,392,393,393,394,394,394,395,395,395,
                                        396,396,396,396,396,397,397,397,397,397,398,400,400

2           J                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        47,51,127,128,134,1148,1148,1148,1150,1151,1153,1153,1157
                                        818,825,826,835,836,842,847,851,853,853,856,889,890,894,894,1113
                                        557,558,608,609,658,659,685,685,686,686,687,687,687,688,688,688,689,692,696,

                                        697,697,697,697,697,698,698,698,698,698,700,700

405         J                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        412,414,414,415,415,430,430,454,454

1115        JUNC_DROP                   ENTRY RETURNS(BINARY FIXED (15,0))
                                        778,822,824,969,1016,1018

543         JUNCTION                    ENTRY RETURNS(BINARY FIXED (15,0))
                                        270

2           K                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        129,130,130,133,134,134,1154,1162,1162,1164

405         K                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        413,437,437,453,453

1116        K                           AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        1119,1126,1129,1129,1129

405         L                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        419,419,426,442,442,449

2           L                           AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        1155,1163,1163,1164

94          LAG                         AUTOMATIC ALIGNED BINARY FIXED (31,0)

********    LAG1                        AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                        95,101,102,104,105

94          LAG2                        AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                        96,97,101,102,104,105

2           LAST_DOWNSTREAM_DATA        AUTOMATIC ALIGNED INITIAL DECIMAL /* SINGLE */ FLOAT (6)
                                        1,535,539
```

```
2          LAST_TRIPLET                AUTOMATIC UNALIGNED BIT (1)
                                       25,213,478,504,535

2          LASTALLOC                   STATIC EXTERNAL ALIGNED POINTER
                                       49,52,148,154,157,163,173,175,176,176,178,190,192,193,193,193,1159,1161,
                                       1168,1171,1172,1172,1188,1190,1193,1195

2          LBFB1                       AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                       118,120,125,129,146,164,167

2          LBFB2                       AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                       122,124,125,133,146,181,184

********   LBOUND                      BUILTIN
                                       42,46,118,122,200,223,223
                                       101,101,101,102,102,102,104,104,105,105
                                       62,77,141,142,258,259,260,261,262,274,275,284,335,336,411,414,478,547,549,
                                       716,1101
                                       599,600,650,651,657,658,708,708,1201,1239,1250,1255,1260,1261,1263,1274,
                                       1293,1295

544        LB3                         AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                       716,722,723,723,724,770,771,775,775,777,779,812,813,817,817,818,821,823,855,
                                       903,903,906,906,910,949,950,954,954,955,961,970,1002,1003,1007,1007,1008,
                                       1008,1019,1019,1020,1020,1043,1046,1047,1048,1048,1051,1084,1085,1089,1089,
                                       1090,1091
                                       553,553

76         LLIM                        AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                       77,80,83,85,86

471        LOGVAL                      AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       485,487,489

********   LOG10                       BUILTIN
                                       95,97
                                       82,353,390,431,455,485,590,641,679,691,1210,1265,1278

2          MANHOLE                     /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                       223,270,553

2          MANHOLE_AREA                /* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       936,937,944,945,989,990,997,998,1023,1070,1071,1078,1079

********   MAX                         BUILTIN
                                       101,102,102,102,102,104,105,105,105
                                       63,143

76         MAXFLOW                     /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       83

94         MAXFLOW                     AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       93,102,102,105,105,106,112

58         MAXI                        AUTOMATIC ALIGNED INITIAL BINARY /* SINGLE */ FLOAT (21)
                                       57,63,63,65,73

56         MAX1                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       91

56         MAX2                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       92

56         MAX3                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

********   MIN                         BUILTIN
                                       125
                                       101,102,102,102,104,105,105,142

********   MOD                         BUILTIN
                                       303

255        NDT                         AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       265,300,386,386,467,467

471        NN                          AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                       474,475,482,486,490,491,491,491,491,491,493,495,496,499,499,499,499,499,500,
                                       502,502,506,507,508,512,513,516,517,523,525,525,528,529,529

2          NODE                        BASED (NODE_PTR) /* STRUCTURE */

2          NODE_#                      /* IN NODE */ BASED ALIGNED BINARY FIXED (16,0)
                                       1144,1151
                                       114,114
                                       70,70,71,71,1311
```

```
********  NODE_PTR                    AUTOMATIC ALIGNED POINTER
                                      35

1229      NODE_PTR                    /* PARAMETER */ ALIGNED POINTER
                                      1230,1231,1232

2         NODE_TYPE                   /* IN NODE */ BASED /* STRUCTURE */

1199      NPT                         /* PARAMETER */ ALIGNED POINTER
                                      1200,1202,1203,1204,1216,1221

76        NPTR                        /* PARAMETER */ ALIGNED POINTER
                                      82,82

2         NULL                        BUILTIN
                                      39,40,45,136,153,213,1191,1196
                                      101
                                      61,402,478,504,541,545,719,1108

1247      NW                          AUTOMATIC /* STRUCTURE */
                                      1299

1247      NWV2                        AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      1277,1283,1284

58        OLD_DIAM                    AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

                                      67,68,70,71,71,71

58        OPTIMAL_DIAMETER            AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      65,68,69,70,71,71,71

********  OUTCON                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                      223,225,226,235,236

2         OUTDPTH                     STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      214

554       OUTFLOW_CONDITION           /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      561,562,572,612,613,623

905       OUTFLOW_CONDITION           AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      906,907,908,909,909,1015,1017,1050

2         OUTVARS                     STATIC EXTERNAL /* STRUCTURE */

2         PAGENO                      STATIC EXTERNAL ALIGNED DECIMAL FIXED (4,0)

1308      PDIAMS                      (0:35) STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      1309,1309,1311,1312

76        PDIAMS                      (0:35) STATIC EXTERNAL ALIGNED INITIAL BINARY /* SINGLE */ FLOAT (21)
                                      75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,
                                      75,75,75,75,75,75,75,75,75,75,75,77,78,81,85,86

2         PIPE_DIAMETER               /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)
                                      95,95,97,97,108,115
                                      67,69,326,407,473,729,736,742,760,766,772,784,790,808,814,827,861,867,885,
                                      891,915,922,929,941,951,957,963,975,982,994,1004,1010,1028,1056,1063,1075,
                                      1086,1092
                                      559,599,602,610,650,653,660,1120,1202,1231,1252,1309,1311,1312

2         PIPE_DROP                   /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)
                                      217,225,235,492,723,723,728,735,748,748,749,749,752,752,753,753,758,764,777,
                                      783,796,797,800,801,806,821,823,855,860,873,874,877,878,883,909,909,921,928,
                                      955,961,981,1008,1015,1017,1050,1062,1090
                                      561,562,572,599,612,613,623,650

2         PIPE_LENGTH                 /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)
                                      3,6,9,95,97

2         PIPE_ROUGHNESS              /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)
                                      95,97

                                      82,353,390,431,455,485,590,641,679,691,1204,1265,1278

2         PIPE_SLOPE                  /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)
                                      95,97
                                      82,356,397,462,499,596,647,682,698,1216,1285

2         PLOTTER                     EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                      1140
```

```
2          PLUCK                    EXTERNAL ENTRY(ALIGNED POINTER ,ALIGNED POINTER ) RETURNS(DECIMAL
                                    /* SINGLE */ FLOAT (6))
                                    49,173,190,1168,1188,1193

58         POINTER                  /* PARAMETER */ ALIGNED POINTER
                                    60,65,66,67,68,69,70,71

1103       POST_PEAK_BASE           AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                    1105,1107,1107,1110,1110,1113

1247       PR                       AUTOMATIC /* STRUCTURE */
                                    1299

2          PRINTERVAL               AUTOMATIC ALIGNED INITIAL BINARY FIXED (15,0)
                                    1,22,303

2          PRTCHG                   EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                    1175

2          PRTFB                    BASED (TFBP) ALIGNED POINTER
                                    163,178,1136,1158
                                    99

2          PRTFB2                   BASED (TFBP2) ALIGNED POINTER
                                    160,193,1137
                                    100

2          PRTFLO                   EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                    263,303

2          PRTIHG                   EXTERNAL ENTRY(ALIGNED POINTER ) RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                    162,180
                                    140

********   PSEUDOBASE               AUTOMATIC ALIGNED POINTER
                                    32,41,44,45,48,52,117,121,148,154,157,161,175,176,178,179,192,193,1138,1158,
                                    1159,1161,1169,1171,1172,1187,1190,1192,1195
                                    98
                                    59,60,61,72,138,139,144,257,281,313,324,325,334,410,477,546,548,1100,1104,

                                     1106,1109,1112
                                     705,706,711,1200,1230,1249

2          PSEUDOREGISTER           BASED (PSEUDOBASE) ALIGNED POINTER
                                    41,44,45,48,52,117,121,148,154,157,161,175,176,178,179,192,193,1138,1158,
                                    1159,1161,1169,1171,1172,1187,1190,1192,1195
                                    98
                                    59,60,61,72,138,139,144,257,281,307,308,309,310,313,324,325,334,410,477,546,
                                    548,1100,1104,1106,1109,1112
                                    705,706,711,1200,1230,1249

319        PT                       /* PARAMETER */ ALIGNED POINTER
                                    321,325,326,338,339,339,353,356,385,387,387,390,397,397

2          PTN#1                    STATIC EXTERNAL ALIGNED POINTER
                                    26

2          PTN#2                    STATIC EXTERNAL ALIGNED POINTER
                                    27

2          PTN#3                    STATIC EXTERNAL ALIGNED POINTER
                                    28

1247       PTR                      /* PARAMETER */ ALIGNED POINTER
                                    1248,1249,1252,1253,1253,1265,1278,1285,1285,1285,1286,1287,1287,1288,1293

1116       PTR                      /* PARAMETER */ ALIGNED POINTER
                                    1117,1120,1121

405        PTR                      /* PARAMETER */ ALIGNED POINTER
                                    406,407,408,410,411,411,411,411,431,437,437,453,453,455,462,466

2          PT1                      /* PARAMETER */ ALIGNED POINTER
                                    3,3,26,35,39,44,117,161,162,162,173,176,195,196,202,202,224,224,225,225,225,
                                    227,230,231,1136,1187,1188,1191
                                    16,91,95,95,95,95,95,97,98,257,266,268,545,723,728,729,730,748,749,752,753,
                                    758,760,761,783,784,785,796,797,800,801,806,808,809,821,822,822,822,909,921,
                                    922,923,955,957,958,981,982,983,1008,1010,1011,1015,1016,1016,1016,1018,
                                    1104
                                    555,555,559,560,561,562,566,567,572,573,590,596,599,599,602,603
```

```
2          PT2                  /* PARAMETER */ ALIGNED POINTER
                                6,6,27,40,41,42,43,49,52,121,160,179,180,180,190,193,197,197,198,203,203,
                                204,204,233,234,234,235,235,235,237,240,241,1137,1192,1193,1196
                                92,92,96,97,97,97,97,99,267,267,269,269,548,718,723,735,736,737,748,749,752,
                                753,764,766,767,777,778,778,778,823,824,824,824,855,860,861,862,873,874,877,
                                878,883,885,886,909,928,929,930,961,963,964,969,969,969,1017,1018,1018,1050,
                                1062,1063,1064,1090,1092,1093,1106
                                606,606,606,610,611,612,613,617,618,623,624,641,647,650,650,653,654

2          PT3                  /* PARAMETER */ ALIGNED POINTER
                                9,9,24,28,136,153,154,199,200,205,205,209,209,210,211,215,217,218,219,223,
                                1138,1144,1151,1168,1169,1172
                                100,101,108,112,114,115,139,140,270,270,272,273,273,281,313,472,473,476,477,
                                484,485,492,499,545,719,721,742,743,747,747,772,773,790,791,795,795,814,815,
                                827,828,829,829,867,868,872,872,891,892,915,916,935,935,936,937,941,942,944,
                                945,951,952,975,976,988,988,989,990,994,995,997,998,1004,1005,1023,1028,
                                1029,1032,1032,1044,1046,1046,1056,1057,1069,1069,1070,1071,1075,1076,1078,
                                1079,1086,1087,1100,1108,1109,1112
                                553,660,661,662,663,663,679,682,691,698,698,706

2          PT4                  AUTOMATIC ALIGNED POINTER
                                24,25,213
                                114,478,504,509,510,511,515,521,521,522,527

1199       Q                    (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1225

405        Q                    (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                414,414,414,414

254        Q_AND_D_VALUES       AUTOMATIC /* STRUCTURE */

********   QB                   AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                1201,1209,1216,1216,1220,1225,1226

1199       QDT                  (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1226

1116       QDT                  (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1132

405        QDT                  (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                465

319        QDT                  (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                381,400

1176       QDT_ROOT             CONTROLLED EXTERNAL /* STRUCTURE */
                                36,1180,1182
                                536

1176       QDTROOT              /* IN QDT_ROOT EXTERNAL */ CONTROLLED ALIGNED BINARY /* SINGLE */ FLOAT
                                (21)
                                38,1177,1181,1181
                                537

2          QDT1                 (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                4,4,5,195,1197


                                266,268,763,770,811,813,822,825,873,877,889,960,1009,1013,1016,1022,1070,
                                1078,1091

2          QDT2                 (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                7,7,8,197,1197
                                267,269,718,769,770,778,796,800,813,824,825,888,889,966,969,989,997,1009,
                                1018,1022,1091,1095

2          QDT3                 (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                10,10,11,199,1197
                                272,274,274,496,502,507,512,517,525,528,529,537,770,775,813,817,825,835,836,
                                847,853,889,894,954,1007,1009,1048,1089,1091,1113
                                700

319        QINF                 AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                332,333,333,364,365,376,381,384

1247       QQB                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1250,1257,1263,1276,1290,1295

254        QVAL                 (2) /* IN Q_AND_D_VALUES */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT
                                (21)
                                259,274,277,277,290,290,290,297,297,299

2          Q1                   (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                4,4,5,195,251,1197
                                268,547,547,564,570
```

```
2          Q2                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 7,7,8,197,251,1197
                                 269,549,549,615,621

2          Q3                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 10,10,11,199,251,1197
                                 259,272,478

********   R                     AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 353,355,390,391,431,432,436,455,456,460,485,486,490,590,591,595,641,642,646,
                                 679,681,691,692,696,1210,1211,1216,1216,1216,1265,1266,1270,1278,1279,1283,
                                 1327

194        RESTART_POINT         /* STATEMENT LABEL CONSTANT */
                                 1314

72         RETURN_POINT          /* STATEMENT LABEL CONSTANT */
                                 68

704        RETVAL                AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 708,709,710,712

471        REY                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 486,488,488

319        REY                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

********   REY                   AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 432,434,434,456,458,458,1211,1213,1214,1266,1268,1268,1279,1281,1281

544        REY                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 591,593,593,642,644,644,692,694,694

76         RLIM                  AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                 78,80,84,86,86

2          ROOT_STATION          /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                 25

319        RSTR                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)

********   RSTR                  AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 433,434,457,458,1212,1213,1267,1268,1280,1281

471        RSTR                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 487,488

544        RSTR                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 592,593,643,644,693,694

137        SAVEPTR               AUTOMATIC ALIGNED POINTER
                                 138,144

704        SAVEPTR               AUTOMATIC ALIGNED POINTER
                                 705,711

288        SAVEVAL               AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 289,290,292

2          SCRCHR                AUTOMATIC UNALIGNED CHARACTER (25) VARYING
                                 20,21,21,21,22

2          SETDIAM               STATIC EXTERNAL UNALIGNED BIT (1)
                                 55

319        SF                    AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 355,356,391,397

1247       SF                    /* IN NW */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 1283,1285

1247       SF                    /* IN PR */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 1270,1285

********   SF                    AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 490,499,595,596,646,647,681,682,696,698

405        SFR                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 436,461,462

405        SFS                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 460,461

********   SIGN                  BUILTIN
                                 1285
```

```
********  SIN                    BUILTIN
                                 376,847,1239,1324,1326

********  SPTR                   AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)

********  SQRT                   BUILTIN
                                 95,97
                                 82,348,423,446,582,633,672,1214,1214,1318,1320,1329

1021      STORG                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 1023,1033,1034,1037,1038

2         SUBSCR1                AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                 125,146,146,149,151,155,156,167,177,184,1141,1142,1154,1156,1160,1162,1163,
                                 1164,1165
                                 142,142

2         SUBSCR2                AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                 130,134,147,147,149,151,156,170,177,187,1142,1144,1147,1148,1151,1155,1157,
                                 1160,1162,1163,1164,1165
                                 143,143

********  SUBSTR                 BUILTIN
                                 20,21

319       SUB1                   AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                 322,323,332,332,332,333,333

319       SUB2                   AUTOMATIC ALIGNED BINARY FIXED (15,0)
                                 323,332

********  SYSPRINT               EXTERNAL FILE PRINT
                                 249,250,251,1144,1151,1178,1179,1181,1184
                                 17,106,109,113,114


                                 70,71,88,283,368,378
                                 328,426,449,757,805,839,849,882,902,903,908,1042,1083
                                 553,585,636,675,1218,1241,1243,1302,1311

2         T                      STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 262,264,264,276,322,332,505,535,539,539,719,1101
                                 710

2         TABLE_PAGE#            /* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0)

319       TC                     AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 385,386,386

405       TC                     AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 466,467,467

2         TEMP                   AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 120,124,124,155,1141,1142,1147,1148
                                 141,141,278,279,280,289,314,316

1199      TEMPPTR                AUTOMATIC ALIGNED POINTER

56        TEMPPTR                AUTOMATIC ALIGNED POINTER
                                 66,70,71

76        TESTDIAM               AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 81,82,82

2         TFB                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 165,168,168,171,171,182,185,185,188,188,1139,1166
                                 101,101,102,102,102,104,104,105,105,105

2         TFBP                   AUTOMATIC ALIGNED POINTER
                                 33,158,163,178,1136,1158
                                 99

2         TFBP2                  AUTOMATIC ALIGNED POINTER
                                 34,159,160,193,1137
                                 100

2         TFB2                   (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                 1139
                                 101,101,102,102,102

********  THETA                  AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                 376,376,753,801,847,847,878,945,998,1038,1038,1079,1239,1239,1316,1318,1320,
                                 1321,1321,1321,1324,1324,1325,1326

2         TI                     STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
```

```
                                95,97,258,262,278,322,332,332,535,1101
                                707,710,710

703         TIME                /* PARAMETER */ ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                707

704         TIMEINC             AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                707,708,709,710,710,710,710

544         TL                  STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1101

319         TPTR                AUTOMATIC ALIGNED POINTER
                                324,334

255         TPTR1               AUTOMATIC ALIGNED POINTER
                                305,307,309

255         TPTR2               AUTOMATIC ALIGNED POINTER
                                306,308,310

255         TPTR3               AUTOMATIC ALIGNED POINTER
                                307,310

255         TPTR4               AUTOMATIC ALIGNED POINTER
                                308,309

405         TR                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                415,422,423,438

********    TR                  AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                574,581,582,625,632,633,664,671,672

2           TREE_CHAIN_POINTERS (3) /* IN NODE */ BASED ALIGNED POINTER
                                24,66,476

319         TS                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                340,347,348

405         TS                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                437,438,445,446

********    TT                  AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                1121,1129

********    TVAL                AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                431,433,435,455,457,459,590,592,594,641,643,645,691,693,695,1265,1267,1269,
                                1278,1280,1282

1199        TVAL                AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1210,1212,1213

405         T1                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                406,415,415,430,430,437,437,453,453,454,454

471         T1                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                484,491,499

544         T1                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                573,574,574,589,589,624,625,625,640,640,662,664,664,680,680,697,697,698,698

319         T1                  AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                338,340,340,354,354,396,396,397,397

318         UPSTRM              ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                266,267

1199        V                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1224

1116        V                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1129,1129,1129,1129

1247        V                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1257,1257,1263,1263,1290,1290,1295,1295

319         V                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                335,340,340,347,354,354,354,354,354,391,392,392,394,394,394,396,396,396,397,
                                397,397,397

1247        V                   /* IN NW */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                1276,1277,1279

405         V                   (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                411,411,415,415,422,430,430,430,430,430,437,437,445,453,453,453,453,453,454,
                                454,454,454,454
```

```
2          VALUATE                 EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                   509

1116       VDT                     (*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   1131

2          VDT1                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   4,4,5,244,1197
                                   266,268,759,763,807,811,822,956,960,1013,1016

2          VDT2                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   7,7,8,246,1197
                                   267,269,765,769,778,824,884,888,962,966,969,1018,1095

2          VDT3                    (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   10,10,11,221,1197
                                   272,495,499,502,508,513,516,525,529,775,817,853,894,950,954,955,961,1003,
                                   1007,1008,1046,1047,1048,1085,1089,1090
                                   698,700

********   VELOCITY1               AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   927,936,937,944,945,987,989,990,997,998

********   VELOCITY2               AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   934,936,937,944,945,1068,1070,1071,1078,1079

********   VELOCITY3               AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   920,921,928,936,937,937,945,980,981,989,990,990,998,1027,1033,1033,1034,
                                   1034,1061,1062,1070,1071,1071,1079

********   VERIFY                  BUILTIN
                                   21

********   VR                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   589,591,595,595,596,640,642,646,646,647

405        VR                      AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   430,432,436,436,461,462

319        VS                      AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   354,355,355,356

405        VS                      AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   453,454,456,460,460,461

********   VS                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   680,681,681,682

********   VV                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   1209,1211,1220,1221,1224

2          V1                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)

                                   4,4,5,195,227,231,244,251,1197
                                   266,268,822,903,1016
                                   553,556,564,569,574,574,581,589,589,589,589,589

2          V2                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   7,7,8,197,237,241,246,251,1197
                                   267,269,778,824,903,969,1018
                                   553,607,615,620,625,625,632,640,640,640,640,640

2          V3                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   10,10,11,199,211,215,219,221,223,223,251,1197
                                   272,474,486,490,491,491,491,491,499,499,499,499,903,906,1020
                                   553,657,664,664,671,680,680,680,680,680,685,685,687,687,687,692,696,697,697,
                                   697,698,698,698,698

29         WORKPRS                 (18) AUTOMATIC ALIGNED POINTER
                                   31,305,306

********   WP                      AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                   1216,1325,1327

1247       X                       /* IN PR */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   1251,1285,1288,1288,1293,1293

1247       X                       /* IN NW */ AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                   1285,1286,1288,1293

703        Y_INTERPOLATOR          ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
                                   719

2          Y_JUNCTION              /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
                                   270,721

2          YBLOCK_POINTER          /* IN NODE */ BASED ALIGNED POINTER
                                   136
                                   100,101,139,545,719,1108,1109
                                   706
```

```
PL/I OPTIMIZING COMPILER          PRTIHG:  PROC(PTR) REORDER;

                    SOURCE LISTING

    STMT LEV NT

      1      0  |PRTIHG:  PROC(PTR) REORDER;   /* PRINTS SPECIFIED INPUT HYDROGRAPHS. */|00000100
      2    1 0  |DCL PAGENO FIXED DEC(4) EXTERNAL,                                     |00000200
                |FLOWBLOCK(*) FLOAT BIN EXTERNAL CONTROLLED,                          |00000300
                |PI FLOAT BIN EXTERNAL,                                               |00000400
                |1 NODE BASED(PTR),                                                   |00001200
                |2 NODE_# FIXED BIN(16),                                              |00001300
                |2 TREE_CHAIN_POINTERS(3) POINTER,                                    |00001400
                |2 DOWNSTREAM_PIPE_INFO,                                              |00001500
                |3 PIPE_LENGTH FIXED BIN(31),                                         |00001600
                |(3 PIPE_SLOPE,                                                       |00001700
                |3 PIPE_DIAMETER,                                                     |00001800
                |3 PIPE_DROP,                                                         |00001900
                |3 PIPE_ROUGHNESS) FLOAT BIN,                                         |00002000
                |3 FUNCTION_INFO,                                                     |00002010
                |4 FUNCPTR POINTER,                                                   |00002020
                |4 FUNCVARA FIXED BIN(15),                                            |00002030
                |4 FUNCVARB FIXED BIN(15),                                            |00002040
                |2 NODE_TYPE,                                                         |00002100
                |3 MANHOLE BIT(1) ALIGNED,                                            |00002200
                |3 Y_JUNCTION BIT(1) ALIGNED,                                         |00002300
                |3 INPUT_STATION BIT(1) ALIGNED,                                      |00002400
                |3 ROOT_STATION BIT(1) ALIGNED,                                       |00002500
                |3 IMAGINARY BIT(1) ALIGNED,                                          |00002600
                |2 FLOWBLOCK_POINTER POINTER,                                         |00002700
                |2 YBLOCK_POINTER POINTER,                                            |00002800
                |2 CRITICAL_DEPTH FLOAT BIN,                                          |00002900
                |2 TABLE_PAGE# FIXED DEC(4),                                          |00003000
                |2 DELTA_X FLOAT BIN,                                                 |00003100
                |2 DNORM FLOAT BIN,                                                   |00003200
                |2 MANHOLE_AREA FLOAT BIN;                                            |00003300
      3    1 0  |DCL PTR POINTER, I;                                                  |00158000
      4    1 0  |SIGNAL ENDPAGE(SYSPRINT);                                            |00158100
      5    1 0  |PTR->TABLE_PAGE#=PAGENO-1;                                           |00158200
      6    1 0  |PUT EDIT('INPUT HYDROGRAPH SPECIFIED FOR NODE #',PTR->NODE_#,(43)'_',|00158300
                |'INCREMENT #   ABS. TIME, SEC   DISCHARGE, CFS',(11)'_'||'   '||    |00158400
                |(14)'_'||'   '||(14)'_',' ')                                         |00158500
                |(SKIP(1),COL(45),A,P'ZZ,Z39',SKIP(0),COL(45),A,SKIP(2),COL(44),     |00158600
                |A,SKIP(0),COL(44),A,SKIP,A)                                         |00158700
                |((I,I*PI,FLOWBLOCK(I) DO I=LBOUND(FLOWBLOCK,1) TO HBOUND(FLOWBLOCK,1)))|00158800
                |(COL(45),P'ZZ,Z49',COL(60),P'ZZZ,ZZ9V.9',COL(78),P'ZZ,ZZ9V.9');      |00158900
      7    1 0  |END PRTIHG;                                                          |00159000
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	CRITICAL_DEPTH	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DELTA_K	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DNORM	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DOWNSTREAM_PIPE_INFO	/* IN NODE */ BASED /* STRUCTURE */
2	FLOWBLOCK	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 6,6,6
2	FLOWBLOCK_POINTER	/* IN NODE */ BASED ALIGNED POINTER
2	FUNCPTR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED POINTER
2	FUNCTION_INFO	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */
2	FUNCVARA	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	FUNCVARR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
********	HBOUND	BUILTIN 6
3	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 6,6,6,6,6
2	IMAGINARY	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	INPUT_STATION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
********	LBOUND	BUILTIN 6
2	MANHOLE	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	MANHOLE_AREA	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	NODE	BASED (PTR) /* STRUCTURE */
2	NODE_TYPE	/* IN NODE */ BASED /* STRUCTURE */
2	PAGENO	STATIC EXTERNAL ALIGNED DECIMAL FIXED (4,0) 5
2	PIPE_DIAMETER	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_DROP	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_LENGTH	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)
2	PIPE_ROUGHNESS	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_SLOPE	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
1	PRTIHG	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
3	PTR	/* PARAMETER */ ALIGNED POINTER 5,6
2	ROOT_STATION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
********	SYSPRINT	EXTERNAL FILE PRINT 4,6
2	TABLE_PAGE#	/* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0) 5
2	TI	STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 6
2	TREE_CHAIN_POINTERS	(3) /* IN NODE */ BASED ALIGNED POINTER
2	Y_JUNCTION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	YBLOCK_POINTER	/* IN NODE */ BASED ALIGNED POINTER

```
PL/I OPTIMIZING COMPILER          PRTFLO:  PROC REORDER;

                    SOURCE LISTING

   STMT LEV NT

     1      0  |PRTFLO:  PROC REORDER;                                                    |00000100
     2    1 0  |DCL                                                                       |00000190
               |(DDT1,DDT2,DDT3,VDT1,VDT2,VDT3,QDT1,QDT2,QDT3)(*)FLOAT BIN EXTERNAL       |00000200
               |CONTROLLED,                                                               |00000210
               |(PTN#2, PTN#1, PTN#3) POINTER EXTERNAL,                                   |00000220
               |(HB1, HB2, HB3) FIXED BIN(15),                                            |00000300
               |IIT FIXED BIN(15),                                                        |00000400
               |TPTR1 POINTER,                                                            |00000500
               |T FLOAT BIN EXTERNAL,                                                     |00000600
               |1 NODE BASED(TPTR1),                                                      |00001200
               |2 NODE_# FIXED BIN(16),                                                   |00001300
               |2 TREE_CHAIN_POINTERS(3) POINTER,                                         |00001400
               |2 DOWNSTREAM_PIPE_INFO,                                                   |00001500
               |3 PIPE_LENGTH FIXED BIN(31),                                              |00001600
               |(3 PIPE_SLOPE,                                                            |00001700
               |3 PIPE_DIAMETER,                                                          |00001800
               |3 PIPE_DROP,                                                              |00001900
               |3 PIPE_ROUGHNESS) FLOAT BIN,                                              |00002000
               |3 FUNCTION_INFO,                                                          |00002010
               |4 FUNCPTR POINTER,                                                        |00002020
               |4 FUNCVAR4 FIXED BIN(15),                                                 |00002030
               |4 FUNCVAR8 FIXED BIN(15),                                                 |00002040
               |2 NODE_TYPE,                                                              |00002100
               |3 MANHOLE BIT(1) ALIGNED,                                                 |00002200
               |3 Y_JUNCTION BIT(1) ALIGNED,                                              |00002300
               |3 INPUT_STATION BIT(1) ALIGNED,                                           |00002400
               |3 ROOT_STATION BIT(1) ALIGNED,                                            |00002500
               |3 IMAGINARY BIT(1) ALIGNED,                                               |00002600
               |2 FLOWBLOCK_POINTER POINTER,                                              |00002700
               |2 YBLOCK_POINTER POINTER,                                                 |00002800
               |2 CRITICAL_DEPTH FLOAT BIN,                                               |00002900
               |2 TABLE_PAGE# FIXED DEC(4),                                               |00003000
               |2 DELTA_X FLOAT BIN,                                                      |00003100
               |2 DNORM FLOAT BIN,                                                        |00003200
               |2 MANHOLE_AREA FLOAT BIN;                                                 |00003300
     3    1 0  |SIGNAL ENDPAGE(SYSPRINT);                                                 |00043500
     4    1 0  |TPTR1=PTN#3->TREE_CHAIN_POINTERS(1);                                      |00043600
     5    1 0  |PUT SKIP(1) EDIT('FLOW CONDITIONS AT TIME =',T,' SECONDS')                |00043700
               |(COL(40),A,P'ZZZ,Z49V.9',A)                                               |00043800
               |('INTERIOR','FROM NODE ',PTN#1->NODE_#,' TO NODE ',PTN#3->NODE_#,         |00043900
               |'FROM NODE ',PTN#2->NODE_#,' TO NODE ',PTN#3->NODE_#,'FROM NODE ',   PT|00044000
               |N#3->NODE_#,' TO NODE ',TPTR1->NODE_#,(31)'_',(31)'_',(31)'_','STATION'|00044100
               |,'NUMBER','VELOCITY   DEPTH   DISCHARGE ','VELOCITY   DEPTH   DISCHARGE|00044200
               | ','VELOCITY   DEPTH   DISCHARGE ',(8)'_',(8)'_',(5)'_',(9)'_',(8)'_',  |00044300
               |(5)'_',(9)'_',(8)'_',(5)'_',(9)'_')                                       |00044400
               |(SKIP(2),COL(10),A,X(3),A,P'ZZ,ZZ9',A,P'ZZ,Z49',X(3),A,P'ZZ,ZZ9',A,     |00044500
               |P'ZZ,ZZ9',X(3),A,P'ZZ,Z49',A,P'ZZ,ZZ9',SKIP(0),COL(18),X(3),A(31),X(3),  |00044600
               |A(31),X(3),A(31),COL(11),A,COL(11),A,X(5),A,X(5),A,X(5),A,SKIP(0),       |00044700
               |COL(10),A,X(4),A,X(3),A,X(3),A,X(2),X(4),A,X(3),A,X(3),A,X(6),A,X(3),    |00044800
               |A,X(3),A);                                                               |00044900
     6    1 0  |HB1=HBOUND(DDT1,1);                                                       |00044910
     7    1 0  |HB2=HBOUND(DDT2,1);                                                       |00044920
     8    1 0  |HB3=HBOUND(DDT3,1);                                                       |00044930
     9    1 0  |DO IIT=0 TO MAX(HB1,HB2,HB3);                                             |00045000
    10    1 1  |PUT SKIP(2) EDIT(IIT)(COL(11),F(4));                                      |00045100
    11    1 1  |IF IIT<=HB1 THEN PUT EDIT(VDT1(IIT),DDT1(IIT),QDT1(IIT))                 |00045200
               |(COL(24),F(5,2),COL(33),F(5,2),COL(43),F(5,1));                           |00045300
    12    1 1  |IF IIT<=HB2 THEN PUT EDIT(VDT2(IIT),DDT2(IIT),QDT2(IIT))                 |00045400
               |(COL(56),F(5,2),COL(67),F(5,2),COL(77),F(5,1));                           |00045500
    13    1 1  |IF IIT<=HB3 THEN PUT EDIT(VDT3(IIT),DDT3(IIT),QDT3(IIT))                 |00045600
               |(COL(92),F(5,2),COL(101),F(5,2),COL(112),F(5,1));                         |00045700
    14    1 1  |END;                                                                      |00045800
    15    1 0  |END PRTFLO;                                                               |00045900
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	CRITICAL_DEPTH	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DDT1	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 6,11
2	DDT2	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 7,12
2	DDT3	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 8,13
2	DELTA_X	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DNORM	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DOWNSTREAM_PIPE_INFO	/* IN NODE */ BASED /* STRUCTURE */
2	FLOWBLOCK_POINTER	/* IN NODE */ BASED ALIGNED POINTER
2	FUNCPTR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED POINTER
2	FUNCTION_INFO	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */
2	FUNCVARA	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	FUNCVARR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
********	HBOUND	BUILTIN 6,7,8
2	HB1	AUTOMATIC ALIGNED BINARY FIXED (15,0) 6,9,11
2	HB2	AUTOMATIC ALIGNED BINARY FIXED (15,0) 7,9,12
2	HB3	AUTOMATIC ALIGNED BINARY FIXED (15,0) 8,9,13
2	IIT	AUTOMATIC ALIGNED BINARY FIXED (15,0) 9,9,10,11,11,11,11,12,12,12,12,13,13,13,13
2	IMAGINARY	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	INPUT_STATION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	MANHOLE	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	MANHOLE_AREA	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
********	MAX	BUILTIN 9
2	NODE	BASED (TPTR1) /* STRUCTURE */
2	NODE_#	/* IN NODE */ BASED ALIGNED BINARY FIXED (16,0) 5,5,5,5,5,5
2	NODE_TYPE	/* IN NODE */ BASED /* STRUCTURE */
2	PIPE_DIAMETER	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_DROP	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_LENGTH	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)
2	PIPE_ROUGHNESS	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_SLOPE	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)

```
1          PRTFLO                    EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          PTN#1                     STATIC EXTERNAL ALIGNED POINTER
                                     5

2          PTN#2                     STATIC EXTERNAL ALIGNED POINTER
                                     5

2          PTN#3                     STATIC EXTERNAL ALIGNED POINTER
                                     4,5,5,5

2          QDT1                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     11

2          QDT2                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)

                                     12

2          QDT3                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     13

2          ROOT_STATION              /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

********   SYSPRINT                  EXTERNAL FILE PRINT
                                     3,5,10,11,12,13

2          T                         STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     5

2          TABLE_PAGE#               /* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0)

2          TPTR1                     AUTOMATIC ALIGNED POINTER
                                     4,5

2          TREE_CHAIN_POINTERS       (3) /* IN NODE */ BASED ALIGNED POINTER
                                     4

2          VDT1                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     11

2          VDT2                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     12

2          VDT3                      (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                     13

2          Y_JUNCTION                /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

2          YBLOCK_POINTER            /* IN NODE */ BASED ALIGNED POINTER
```

SOURCE LISTING

```
STMT LEV NT

   1      0  |PRTCHG:  PROC REORDER;  /* PRINTS COMPUTED HYDROGRAPHS. */                |00000100
   2    1 0  |DCL (FLOWBLOCK,TFB,TFB2,DEPTHBLOCK) (*) FLOAT BIN EXTERNAL CTL,          |00000200
             |TI FLOAT BIN EXTERNAL,                                                   |00000300
             |(PAGENO EXTERNAL, K) FIXED DEC(4),                                       |00000400
             |SOPARM CHAR(20) EXTERNAL VAR,                                            |00000500
             |(PTN#1, PTN#2, PTN#3) POINTER EXTERNAL,                                  |00000600
             |1 NODE BASED(PTN#1),                                                     |00001200
             |2 NODE_# FIXED BIN(16),                                                  |00001300
             |2 FREE_CHAIN_POINTERS(3) POINTER,                                        |00001400
             |2 DOWNSTREAM_PIPE_INFO,                                                  |00001500
             |3 PIPE_LENGTH FIXED BIN(31),                                             |00001600
             |(3 PIPE_SLOPE,                                                           |00001700
             |3 PIPE_DIAMETER,                                                         |00001800
             |3 PIPE_DROP,                                                             |00001900
             |3 PIPE_ROUGHNESS) FLOAT BIN,                                             |00002000
             |3 FUNCTION_INFO,                                                         |00002010
             |4 FUNCPTR POINTER,                                                       |00002020
             |4 FUNCVARA FIXED BIN(15),                                                |00002030
             |4 FUNCVARB FIXED BIN(15),                                                |00002040
             |2 NODE_TYPE,                                                             |00002100
             |3 MANHOLE BIT(1) ALIGNED,                                                |00002200
             |3 Y_JUNCTION BIT(1) ALIGNED,                                             |00002300
             |3 INPUT_STATION BIT(1) ALIGNED,                                          |00002400
             |3 ROOT_STATION BIT(1) ALIGNED,                                           |00002500
             |3 IMAGINARY BIT(1) ALIGNED,                                              |00002600
             |2 FLOWBLOCK_POINTER POINTER,                                             |00002700
             |2 YBLOCK_POINTER POINTER,                                                |00002800
             |2 CRITICAL_DEPTH FLOAT BIN,                                              |00002900
             |2 TABLE_PAGE# FIXED DEC(4),                                              |00003000
             |2 DELTA_X FLOAT BIN,                                                     |00003100
             |2 DNORM FLOAT BIN,                                                       |00003200
             |2 MANHOLE_AREA FLOAT BIN;                                                |00003300
   3    1 0  |SIGNAL ENDPAGE(SYSPRINT);                                                |00135900
   4    1 0  |K=PAGENO-1;                                                              |00136000
   5    1 0  |PUT EDIT('INPUT HYDROGRAPH CALCULATED FOR NODE #',PTN#3->NODE_#,         |00136100
             |(44)'_',                                                                 |00136110
             |'ABS. TIME, SEC   DEPTH, FEET   DISCHARGE, CFS',(14)'_'||'   '||(11)'_'|00136200
             |||'   '||(14)'_')                                                        |00136300
             |(SKIP(1),COL(45),A,P'ZZ,ZZ9',SKIP(0),COL(45),A,SKIP(2),COL(44),A,       |00136400
             |SKIP(0),COL(44),A)                                                      |00136500
             |((I*TI,DEPTHBLOCK(I),FLOWBLOCK(I) DO I=LBOUND(FLOWBLOCK,1) TO HBOUND(   |00136600
             |FLOWBLOCK,1)))                                                          |00136700
             |(COL(45),P'ZZZ,ZZ9V.9',COL(64),F(5,2),COL(78),P'ZZ,ZZ9V.9');            |00136800
   6    1 0  |PUT SKIP(2) EDIT((47)'_','|','PREVIOUS HYDROGRAPHS','|',(47)'_',        |00136900
             |'|','NODE','NUMBER','SEE PAGE','|',(4)'_',(6)'_',(8)'_','|',           |00137000
             |' FIRST UPSTREAM',PTN#1->NODE_#,PTN#1->TABLE_PAGE#,'|')                 |00137100
             |(COL(43),A,COL(42),A,X(13),A,X(14),A,SKIP(0),COL(43),A,COL(42),A,       |00137200
             |X(10),A,COL(67),A,COL(78),A,COL(90),A,SKIP(0),COL(53),A,COL(67),A,COL   |00137300
             |(78),A,R(OTHER_HYDROGRAPHS));                                           |00137400
   7    1 0  |IF PTN#2->TABLE_PAGE#¬=0 THEN PUT EDIT('|','SECOND UPSTREAM',PTN#2->    |00137500
             |NODE_#, PTN#2->TABLE_PAGE#,'|') (R(OTHER_HYDROGRAPHS));                  |00137600
   8    1 0  |IF PTN#3->TABLE_PAGE#¬=0 THEN PUT EDIT('|','JUNCTION INFLOW',PTN#3->    |00137700
             |NODE_#,PTN#3->TABLE_PAGE#,'|') (R(OTHER_HYDROGRAPHS));                  |00137800
   9    1 0  |OTHER_HYDROGRAPHS:  FORMAT(COL(42),A,COL(47),A,COL(67),P'ZZ,ZZ9',      |00137900
             |COL(90),F(4),COL(90),A);                                                |00138000
  10    1 0  |PUT EDIT((47)'_') (SKIP(0),COL(43),A);                                  |00138100
  11    1 0  |PTN#3->TABLE_PAGE#=K;                                                    |00138200
  12    1 0  |END PRTCHG;                                                              |00138250
```

PL/I OPTIMIZING COMPILER PRTCHG: PROC REORDER; /* PRINTS COMPUTED HYDROGRAPHS. */

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	CRITICAL_DEPTH	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DELTA_X	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DEPTHBLOCK	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 5
2	DNORM	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DOWNSTREAM_PIPE_INFO	/* IN NODE */ BASED /* STRUCTURE */
2	FLOWBLOCK	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 5,5,5
2	FLOWBLOCK_POINTER	/* IN NODE */ BASED ALIGNED POINTER
2	FUNCPTR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED POINTER
2	FUNCTION_INFO	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */
2	FUNCVAR1	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	FUNCVAR2	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	GOPARM	STATIC EXTERNAL UNALIGNED CHARACTER (20) VARYING
********	HBOUND	BUILTIN 5
********	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 5,5,5,5,5
2	IMAGINARY	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	INPUT_STATION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	K	AUTOMATIC ALIGNED DECIMAL FIXED (4,0) 4,11
********	LBOUND	BUILTIN 5
2	MANHOLE	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	MANHOLE_AREA	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	NODE	BASED (PTR#1) /* STRUCTURE */
2	NODE_#	/* IN NODE */ BASED ALIGNED BINARY FIXED (16,0) 5,6,7,8
2	NODE_TYPE	/* IN NODE */ BASED /* STRUCTURE */
9	OTHER_HYDROGRAPHS	/* STATEMENT LABEL CONSTANT */ 6,7,8
2	PAGENO	STATIC EXTERNAL ALIGNED DECIMAL FIXED (4,0) 4
2	PIPE_DIAMETER	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_DROP	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	PIPE_LENGTH	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)

```
2          PIPE_ROUGHNESS              /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                       FLOAT (21)

2          PIPE_SLOPE                  /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                       FLOAT (21)

1          PRTCHG                      EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          PTN#1                       STATIC EXTERNAL ALIGNED POINTER
                                       6,6

2          PTN#2                       STATIC EXTERNAL ALIGNED POINTER
                                       7,7,7

2          PTN#3                       STATIC EXTERNAL ALIGNED POINTER
                                       5,8,8,8,11

2          ROOT_STATION                /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

********   SYSPRINT                    EXTERNAL FILE PRINT
                                       3,5,6,7,8,10

2          TABLE_PAGE#                 /* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0)
                                       6,7,7,8,8,11

2          TF3                         (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)

2          TF32                        (*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)

2          TI                          STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                       5

2          TREE_CHAIN_POINTERS         (3) /* IN NODE */ BASED ALIGNED POINTER

2          Y_JUNCTION                  /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

2          YBLOCK_POINTER              /* IN NODE */ BASED ALIGNED POINTER
```

```
PL/I OPTIMIZING COMPILER          GRAPHER:  PROC(A1,A2,A3) REORDER;

                    SOURCE LISTING

    STMT LEV NT

      1       0  |GRAPHER:  PROC(A1,A2,A3) REORDER;                                    |00145500
      2   1   0  |DCL PAGE( 0 :28,-14:105) CHAR(1) INIT((3480)(1)' '),                 |00145600
                 |(MAX3(3),(A1,A2,A3)(*),HINC,VINC) FLOAT BIN,                        |00145700
                 |GRAPHFL BIT(1) EXTERNAL,                                            |00145710
                 |HPOS FIXED BIN(15);                                                 |00145800
      3   1   0  |ON ERROR SNAP BEGIN;                                                |00145900
      4   2   0  |PUT SKIP(2) EDIT('*GRAPH SUBROUTINE ERROR.  GRAPHER IS BEING DISABLED.'|00146000
                 |)(A);                                                               |00146100
      5   2   0  |GRAPHFL='0'B;     /* SO GRAPHER WON'T BE CALLED AGAIN. */           |00146200
      6   2   0  |GOTO GRAPH_END;                                                     |00146300
      7   2   0  |END;                                                                |00146400
      8   1   0  |MAX3(1)=A1(LBOUND(A1,1));                                           |00146500
      9   1   0  |MAX3(2)=A2(LBOUND(A2,1));                                           |00146600
     10   1   0  |MAX3(3)=A3(LBOUND(A3,1));                                           |00146700
     11   1   0  |DO I=LBOUND(A1,1)+1 TO HBOUND(A1,1);                                |00146800
     12   1   1  |MAX3(1)=MAX(MAX3(1),A1(I));                                         |00146900
     13   1   1  |MAX3(2)=MAX(MAX3(2),A2(I));                                         |00147000
     14   1   1  |MAX3(3)=MAX(MAX3(3),A3(I));                                         |00147100
     15   1   1  |END;                                                                |00147200
     16   1   0  |VINC=MAX(MAX3(1),MAX3(2),MAX3(3))/27;                               |00147300
     17   1   0  |JADJ=DIM(A1,1);                                                     |00147400
     18   1   0  |HINC=JADJ/33;                                                       |00147500
     19   1   0  |GRAPHINIT:  DO I=1,105;                                             |00147600
     20   1   1  |DO J=0 TO 27;                                                       |00147700
     21   1   2  |PAGE(J,I)='|';   /* PUT VERTICAL BARS DOWN BOTH SIDES. */          |00147800
     22   1   2  |END; END;                                                           |00147900
     24   1   0  |DO I=0,28;                                                          |00148000
     25   1   1  |DO J=2 TO 104;                                                      |00148100
     26   1   2  |PAGE(I,J)='_';  /* PUT UNDERBARS AT TOP AND BOTTOM. */              |00148200
     27   1   2  |END; END;                                                           |00148300
     29   1   0  |DO I=LBOUND(A1,1) TO HBOUND(A1,1);                                  |00148400
     30   1   1  |HPOS=3*(I-LBOUND(A1,1))/HINC+2;                                     |00148500
     31   1   1  |PAGE(A1(I)/VINC,HPOS+1)='1';                                        |00148600
     32   1   1  |PAGE(A2(I)/VINC,HPOS+2)='2';                                        |00148700
     33   1   1  |PAGE(A3(I)/VINC,HPOS+3)='3';                                        |00148800
     34   1   1  |END;                                                                |00148900
     35   1   0  |SIGNAL ENDPAGE(SYSPRINT);                                           |00149000
     36   1   0  |PUT LINE(15);                                                       |00149100
     37   1   0  |DO I=28 TO 0 BY -1;                                                 |00149200
     38   1   1  |PUT EDIT(PAGE(I,*))(A);                                             |00149300
     39   1   1  |END;                                                                |00149400
     40   1   0  |GRAPH_END:                                                          |00149500
                 |END GRAPHER;                                                        |00149600
```

PL/I OPTIMIZING COMPILER GRAPHER: PROC(A1,A2,A3) REORDER;

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	A1	(*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21) 8,8,11,11,12,17,29,29,30,31
2	A2	(*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21) 9,9,13,32
2	A3	(*) /* PARAMETER */ ALIGNED BINARY /* SINGLE */ FLOAT (21) 10,10,14,33
********	DIM	BUILTIN 17
40	GRAPH_2ND	/* STATEMENT LABEL CONSTANT */ 6
1	GRAPHER	EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))
2	GRAPHFL	STATIC EXTERNAL UNALIGNED BIT (1) 5
19	GRAPHINIT	/* STATEMENT LABEL CONSTANT */
********	HBOUND	BUILTIN 11,29
2	HINC	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 18,30
2	HPOS	AUTOMATIC ALIGNED BINARY FIXED (15,0) 30,31,32,33
********	I	AUTOMATIC ALIGNED BINARY FIXED (15,0) 11,11,12,13,14,19,19,19,19,21,24,24,24,24,25,29,29,30,31,32,33,37,37,38
********	J	AUTOMATIC ALIGNED BINARY FIXED (15,0) 20,20,21,25,25,26
********	JADJ	AUTOMATIC ALIGNED BINARY FIXED (15,0) 17,18
********	LBOUND	BUILTIN 8,9,10,11,29,30
********	MAX	BUILTIN 12,13,14,16
2	MAX3	(3) AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 8,9,10,12,12,13,13,14,14,16,16,16
2	PAGE	(0:28,-14:105) AUTOMATIC UNALIGNED INITIAL CHARACTER (1) 1,21,26,31,32,33,38
********	SYSPRINT	EXTERNAL FILE PRINT 35,36,38 4
2	VINC	AUTOMATIC ALIGNED BINARY /* SINGLE */ FLOAT (21) 16,31,32,33

SOURCE LISTING

```
STMT LEV NT

   1      0  |PLOTTER:  PROC REORDER;                                                          |00000100
   2    1 0  |DCL (DEPTHBLOCK, FLOWBLOCK)(*) FLOAT BIN EXTERNAL CONTROLLED,                    |00000200
             |GOPARM CHAR(20) VAR EXTERNAL,                                                    |00000300
             |CALCOMP OUTPUT RECORD SEQUENTIAL ENV(VBS(7294,10000) BUFFERS(1)),                |00000400
             |K FIXED DEC(4),                                                                 |00000600
             |TI FLOAT BIN EXTERNAL,                                                          |00000800
             |(PTN#1, PTN#2, PTN#3) POINTER EXTERNAL,                                         |00001000
             |1 NODE BASED(PTN#1),                                                            |00001200
             |2 NODE_# FIXED BIN(16),                                                         |00001300
             |2 TREE_CHAIN_POINTERS(3) POINTER,                                               |00001400
             |2 DOWNSTREAM_PIPE_INFO,                                                         |00001500
             |3 PIPE_LENGTH FIXED BIN(31),                                                    |00001600
             |(3 PIPE_SLOPE,                                                                  |00001700
             |3 PIPE_DIAMETER,                                                                |00001800
             |3 PIPE_DROP,                                                                    |00001900
             |3 PIPE_ROUGHNESS) FLOAT BIN,                                                    |00002000
             |3 FUNCTION_INFO,                                                                |00002010
             |4 FUNCPTR POINTER,                                                              |00002020
             |4 FUNCVARA FIXED BIN(15),                                                       |00002030
             |4 FUNCVARR FIXED BIN(15),                                                       |00002040
             |2 NODE_TYPE,                                                                    |00002100
             |3 MANHOLE BIT(1) ALIGNED,                                                       |00002200
             |3 Y_JUNCTION BIT(1) ALIGNED,                                                    |00002300
             |3 INPUT_STATION BIT(1) ALIGNED,                                                 |00002400
             |3 NODE_STATION BIT(1) ALIGNED,                                                  |00002500
             |3 IMAGINARY BIT(1) ALIGNED,                                                     |00002600
             |2 FLOWBLOCK_POINTER POINTER,                                                    |00002700
             |2 YBLOCK_POINTER POINTER,                                                       |00002800
             |2 CRITICAL_DEPTH FLOAT BIN,                                                     |00002900
             |2 TABLE_PAGES FIXED DEC(4),                                                     |00003000
             |2 DELTA_X FLOAT BIN,                                                            |00003100
             |2 DNORM FLOAT BIN,                                                              |00003200
             |2 MANHOLE_AREA FLOAT BIN;                                                       |00003300
   3    1 0  |ON ERROR SNAP GOTO RETURN;                                                      |00129000
   4    1 0  |ON UNDEFINEDFILE(CALCOMP) BEGIN;                                                |00129100
   5    2 0  |PUT EDIT('** ERROR **  THE "//CALCOMP DD  " CARD IS MISSING IN THE JCL.         |00129200
             |  "PLOT" OPTION IS BEING DISABLED.')(SKIP(3),A);                                 |00129300
   6    2 0  |SUBSTR(GOPARM,INDEX(GOPARM,'PLOT'),4)='    ';                                   |00129400
   7    2 0  |GOTO RETURN;  END;                                                              |00129500
   8    1 0  |DCL 1 DESCRIPTOR,                                                               |00129600
             |2 CHECKER CHAR(24) INIT('RECORD DESCRIPTOR RECORD'),                            |00129700
             |(2 HB, 2 LB, 2 TINC, 2 NN1, 2 NN2, 2 NN3) FIXED BIN(31);                       |00129800
  10    1 0  |DESCRIPTOR.HB=HBOUND(FLOWBLOCK,1);                                              |00129900
  11    1 0  |DESCRIPTOR.LB=LBOUND(FLOWBLOCK,1);                                              |00130000
  12    1 0  |DESCRIPTOR.TINC=TI;                                                             |00130100
  13    1 0  |DESCRIPTOR.NN1=PTN#1->NODE_#;                                                   |00130200
  14    1 0  |DESCRIPTOR.NN2=PTN#2->NODE_#;                                                   |00130300
  15    1 0  |DESCRIPTOR.NN3=PTN#3->NODE_#;                                                   |00130400
  16    1 0  |WRITE FILE(CALCOMP) FROM(DESCRIPTOR);                                          |00130500
  17    1 0  |WRITE FILE(CALCOMP) FROM(FLOWBLOCK);                                           |00130600
  18    1 0  |WRITE FILE(CALCOMP) FROM(TFB2);                                                |00130700
  19    1 0  |WRITE FILE(CALCOMP) FROM(TFB);                                                 |00130800
  20    1 0  |WRITE FILE(CALCOMP) FROM(DEPTHBLOCK);                                          |00130900
  21    1 0  |RETURN:  END PLOTTER;                                                           |00131000
```

PL/I OPTIMIZING COMPILER PLOTTER: PROC REORDER;

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2	CALCOMP	EXTERNAL FILE RECORD SEQUENTIAL OUTPUT ENVIRONMENT(VBS(7294,10000) BUFFERS(1)) 4,16,17,18,19,20
9	CHECKER	/* IN DESCRIPTOR */ AUTOMATIC UNALIGNED INITIAL CHARACTER (24) 1
2	CRITICAL_DEPTH	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DELTA_X	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DEPTHBLOCK	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 20
9	DESCRIPTOR	AUTOMATIC /* STRUCTURE */ 16
2	DNORM	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
2	DOWNSTREAM_PIPE_INFO	/* IN NODE */ BASED /* STRUCTURE */
2	FLOWBLOCK	(*) CONTROLLED EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21) 10,11,17
2	FLOWBLOCK_POINTER	/* IN NODE */ BASED ALIGNED POINTER
2	FUNCPTR	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED POINTER
2	FUNCTION_INFO	/* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED /* STRUCTURE */
2	FUNCVARA	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	FUNCVARB	/* IN FUNCTION_INFO IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (15,0)
2	GOPARM	STATIC EXTERNAL UNALIGNED CHARACTER (20) VARYING 6,6
9	HB	/* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0) 10
********	HBOUND	BUILTIN 10
2	IMAGINARY	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
********	INDEX	BUILTIN 6
2	INPUT_STATION	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	K	AUTOMATIC ALIGNED DECIMAL FIXED (4,0)
9	LB	/* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0) 11
********	LBOUND	BUILTIN 11
2	MANHOLE	/* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)
2	MANHOLE_AREA	/* IN NODE */ BASED ALIGNED BINARY /* SINGLE */ FLOAT (21)
9	NN1	/* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0) 13
9	NN2	/* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0) 14
9	NN3	/* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0) 15
2	NODE	BASED (PTN#1) /* STRUCTURE */
2	NODE_#	/* IN NODE */ BASED ALIGNED BINARY FIXED (16,0) 13,14,15

```
2          NODE_TYPE                  /* IN NODE */ BASED /* STRUCTURE */

2          PIPE_DIAMETER              /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)

2          PIPE_DROP                  /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)

2          PIPE_LENGTH                /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY FIXED (31,0)

2          PIPE_ROUGHNESS             /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)

2          PIPE_SLOPE                 /* IN DOWNSTREAM_PIPE_INFO IN NODE */ BASED ALIGNED BINARY /* SINGLE */
                                      FLOAT (21)

1          PLOTTER                    EXTERNAL ENTRY RETURNS(DECIMAL /* SINGLE */ FLOAT (6))

2          PTR#1                      STATIC EXTERNAL ALIGNED POINTER
                                      13

2          PTR#2                      STATIC EXTERNAL ALIGNED POINTER
                                      14

2          PTR#3                      STATIC EXTERNAL ALIGNED POINTER
                                      15

21         RETURN                     /* STATEMENT LABEL CONSTANT */
                                      3,7

2          ROOT_STATION               /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

********   SUBSTR                     BUILTIN
                                      6

********   SYSPRINT                   EXTERNAL FILE PRINT
                                      5

2          TABLE_PAGE#                /* IN NODE */ BASED ALIGNED DECIMAL FIXED (4,0)

********   TPB                        AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                      19

********   TPB2                       AUTOMATIC ALIGNED DECIMAL /* SINGLE */ FLOAT (6)
                                      18

2          TI                         STATIC EXTERNAL ALIGNED BINARY /* SINGLE */ FLOAT (21)
                                      12

9          TINC                       /* IN DESCRIPTOR */ AUTOMATIC ALIGNED BINARY FIXED (31,0)
                                      12

2          TREE_CHAIN_POINTERS        (3) /* IN NODE */ BASED ALIGNED POINTER

2          Y_JUNCTION                 /* IN NODE_TYPE IN NODE */ BASED ALIGNED BIT (1)

2          YBLOCK_POINTER             /* IN NODE */ BASED ALIGNED POINTER
```

APPENDIX C

Plotting Program Using Calcomp Plotter

The Calcomp Plotter program used for plotting the discharge and depth graphs in the computer system of the University of Illinois at Urbana-Champaign is listed in the following pages. As it has been stated in Subsection 5.2.2, most computer installations have made some modifications to the basic CALCOMP subroutines and modifications should be made accordingly.

```
      GRAFIT:  PROC OPTIONS(MAIN);

STMT LEVEL NEST
  1              GRAFIT:  PROC OPTIONS(MAIN);
  2     1        DCL CCP5AX ENTRY(FLOAT,FLOAT,CHAR(*),FIXED BIN(31),FLOAT,FLOAT,
                 (2) FLOAT),
                 LINE8Z ENTRY((*) FLOAT, (*) FLOAT, FIXED BIN(31), FIXED BIN(31),
                 (2) FLOAT, (2) FLOAT, FIXED BIN(31), FIXED BIN(31)),
                 PLOT ENTRY(FLOAT,FLOAT,FIXED BIN(31)),
                 SYMBOL ENTRY(FLOAT,FLOAT,FLOAT,,FLOAT,FIXED BIN(31)),
                 (TSCALE,DISCALE,DESCALE)(2) FLOAT INIT((2)0),
                 (#PTS1,#PTS2,#PTS3,#PTS4,#PTS5,#PTS6) FIXED BIN(31),
                 (XN1,XN2,XN3,XN4,XN5,XN6,YN1,YN2,YN3,YN4,YN5,YN6)(50) FLOAT,
                 (NODE1#,NODE2#,NODE3#) PIC'ZZ,ZZ9',
                 (ONE INIT(1), TWO INIT(2), ZERO INIT(0)) FIXED BIN(31);
  3     1        ON ENDFILE(SYSIN) GOTO DONE;
  5     1        CALL PLOT(2.0,2.7,-3);  /* MOVE OFF OF THE BASE LINE */
  6     1        L=2;
  7     1        DRAW_SEPARATOR:  CALL PLOT(-1.9,0,3); CALL PLOT(-1.9,7,2);
  9     1        GET DATA(TSCALE,DISCALE,DESCALE,#PTS1,#PTS2,#PTS3,#PTS4,#PTS5,#PTS6,
                 NODE1#,NODE2#,NODE3#,L);
 10     1        DO I = 1 TO #PTS1;  GET LIST(XN1(I),YN1(I));  END;
 13     1        DO I = 1 TO #PTS2;  GET LIST(XN2(I),YN2(I));  END;
 16     1        DO I = 1 TO #PTS3;  GET LIST(XN3(I),YN3(I));  END;
 19     1        DO I = 1 TO #PTS4;  GET LIST(XN4(I),YN4(I));  END;
 22     1        DO I = 1 TO #PTS5;  GET LIST(XN5(I),YN5(I));  END;
 25     1        DO I = 1 TO #PTS6;  GET LIST(XN6(I),YN6(I));  END;
 28     1        DISCHARGE_LOWER_GRAPH:
                 CALL CCP5AX(0.,0.,'TIME, MIN',-30,5.5,0,TSCALE);  /* X-AXIS */
 29     1        CALL CCP5AX(0.,0.,'DISCHARGE, CFS',30,3.1,90,DISCALE);  /* Y-AXIS */
 30     1        IF #PTS1¬=0 THEN
 31     1        CALL LINE8Z(XN1,YN1,#PTS1,1,TSCALE,DISCALE,#PTS1/L,0);
 32     1        IF #PTS2¬=0 THEN
 33     1        CALL LINE8Z(XN2,YN2,#PTS2,1,TSCALE,DISCALE,#PTS2/L,1);
 34     1        IF #PTS3¬=0 THEN
 35     1        CALL LINE8Z(XN3,YN3,#PTS3,1,TSCALE,DISCALE,#PTS3/L,2);
 36     1        DEPTH_UPPER_GRAPH:
                 CALL PLOT(0.,4.,-3);
 37     1        CALL CCP5AX(0.,0.,'TIME, MIN',-30,5.5,0,TSCALE);  /* X-AXIS */
 38     1        CALL CCP5AX(0.,3.,'DEPTH, FEET',90,3.1,90,DESCALE);
 39     1        IF #PTS4¬=0 THEN
 40     1        CALL LINE8Z(XN4,YN4,#PTS4,1,TSCALE,DESCALE,#PTS4/L,0);
 41     1        IF #PTS5¬=0 THEN
 42     1        CALL LINE8Z(XN5,YN5,#PTS5,1,TSCALE,DESCALE,#PTS5/L,1);
 43     1        IF #PTS6¬=0 THEN
 44     1        CALL LINE8Z(XN6,YN6,#PTS6,1,TSCALE,DESCALE,#PTS6/L,2);
 45     1        NOW_THE_BOX:
 46     1        CALL PLOT(3.5,3,3); CALL PLOT(5.5,3,2); CALL PLOT(5.5,2,2);
 48     1                            CALL PLOT(3.5,2,2); CALL PLOT(3.5,3,2);
 50     1        CALL PLOT(3.44,3.06,3); CALL PLOT(5.56,3.06,2); CALL PLOT(5.56,1.94,2);
 53     1                                CALL PLOT(3.44,1.94,2); CALL PLOT(3.44,3.06,2);
 55     1        CALL SYMBOL(3.9,2.1875,.125,'- NODE '||NODE1#,0.,13);
 56     1        CALL SYMBOL(3.9,2.4375,.125,'- NODE '||NODE2#,0.,13);
 57     1        CALL SYMBOL(3.9,2.6875,.125,'- NODE '||NODE3#,0.,13);
 58     1        CALL SYMBOL(3.7,2.25,.1875,ZERO,0.,-1);
 59     1        CALL SYMBOL(3.7,2.5,.1875,ONE,0.,-1);
 60     1        CALL SYMBOL(3.7,2.75,.1875,TWO,0.,-1);
 61     1        ADVANCE:  CALL PLOT(8.5,-4.,-3);
 62     1        GOTO DRAW_SEPARATOR;
 63     1        DONE:  END GRAFIT;
```

```
GRAFIT:  PROC OPTIONS(MAIN);
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
61	ADVANCE	STATEMENT LABEL CONSTANT
2	CCP5AX	EXTERNAL,ENTRY,DECIMAL,FLOAT(SINGLE) 28,29,37,38
36	DEPTH_UPPER_GRAPH	STATEMENT LABEL CONSTANT
2	DESCALE	(2) AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FLOAT(SINGLE) 9,38,40,42,44
2	DISCALE	(2) AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FLOAT(SINGLE) 9,29,31,33,35
28	DISCHARGE_LOWER_GRAPH	STATEMENT LABEL CONSTANT
63	DONE	STATEMENT LABEL CONSTANT 4
7	DRAW_SEPARATOR	STATEMENT LABEL CONSTANT 62
1	GRAFIT	ENTRY,DECIMAL,FLOAT(SINGLE)
*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 10,11,11,13,14,14,16,17,17,19,20,20,22,23,23,25,25,26
2	#PTS1	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,10,30,31,31
2	#PTS2	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,13,32,33,33
2	#PTS3	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,16,34,35,35
2	#PTS4	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,19,39,40,40
2	#PTS5	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,22,41,42,42
2	#PTS6	AUTOMATIC,ALIGNED,BINARY,FIXED(31,0) 9,25,43,44,44
*********	L	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 6,9,31,33,35,40,42,44
2 *********	LINE8Z	EXTERNAL,ENTRY,BINARY,FIXED(15,0) 31,33,35,40,42,44
2	NODE1#	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(ZZ,ZZ9) 9,55
2	NODE2#	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(ZZ,ZZ9) 9,56
2	NODE3#	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(ZZ,ZZ9) 9,57
45	NOW_THE_BOX	STATEMENT LABEL CONSTANT
2	ONE	AUTOMATIC,ALIGNED,INITIAL,BINARY,FIXED(31,0) 59
2	PLOT	EXTERNAL,ENTRY,DECIMAL,FLOAT(SINGLE) 5,7,8,36,45,46,47,48,49,50,51,52,53,54,61
2	SYMBOL	EXTERNAL,ENTRY,DECIMAL,FLOAT(SINGLE) 55,56,57,58,59,60
	SYSIN	FILE,EXTERNAL 3,9,11,14,17,20,23,26
2	TSCALE	(2) AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FLOAT(SINGLE) 9,28,31,33,35,37,40,42,44

```
2          TWO          AUTOMATIC,ALIGNED,INITIAL,BINARY,FIXED(31,0)
                        60

2          XN1          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        11,31

2          XN2          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        14,33

2          XN3          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        17,35

2          XN4          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        20,40

2          XN5          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        23,42

2          XN6          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        26,44

2          YN1          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        11,31

2          YN2          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        14,33

2          YN3          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        17,35

2          YN4          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        20,40

2          YN5          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        23,42

2          YN6          (50)AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE)
                        26,44

2          ZERO         AUTOMATIC,ALIGNED,INITIAL,BINARY,FIXED(31,0)
                        58
```